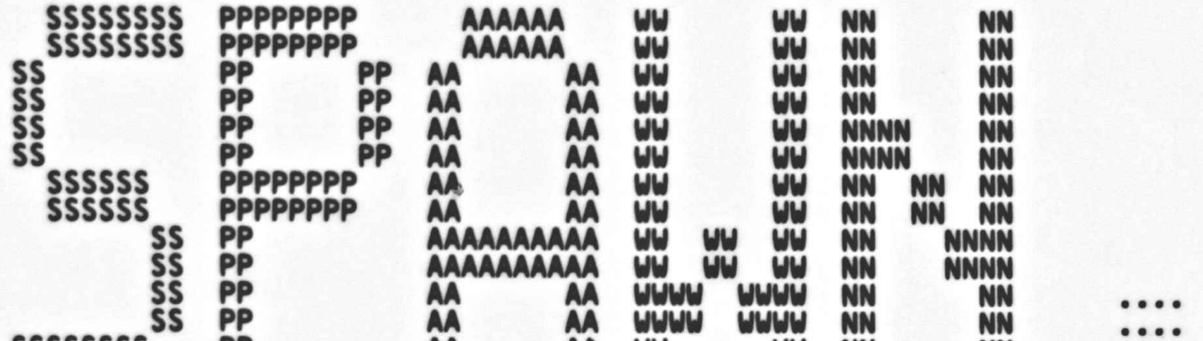
```
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
```


SPAWN

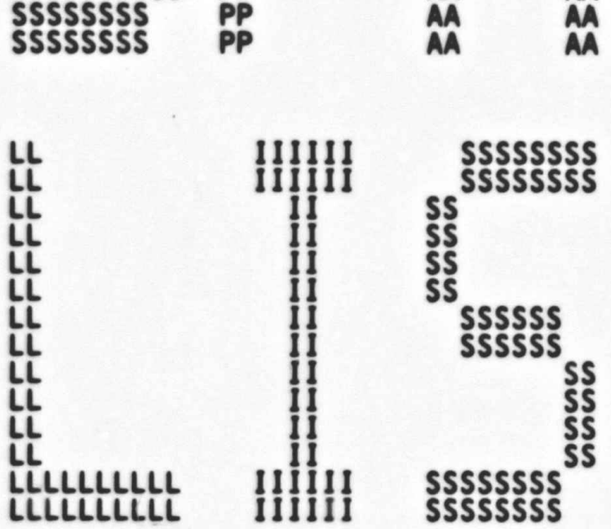

LISS

Table of contents

```
0000     1
0000     2          .TITLE  SPAWN - MULTI-PROCESSING COMMANDS
0000     3          .IDENT  'V04-000'
0000     4 ;
0000     5 ;****************************************************************************
0000     6 ;*                                                                          *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     9 ;*  ALL RIGHTS RESERVED.                                                    *
0000    10 ;*                                                                          *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    16 ;*  TRANSFERRED.                                                            *
0000    17 ;*                                                                          *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    20 ;*  CORPORATION.                                                            *
0000    21 ;*                                                                          *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    24 ;*                                                                          *
0000    25 ;*                                                                          *
0000    26 ;****************************************************************************
0000    27 ;
0000    28 ; AUTHOR:
0000    29 ;
0000    30 ;       Tim Halvorsen, May 1981
0000    31 ;
0000    32 ; MODIFIED BY:
0000    33 ;
0000    34 ;       V03-025 SSA0029         Stan Amway                  6-Aug-1984
0000    35 ;               Propogate WS default correctly. GETJPI item code was
0000    36 ;               JPI$_WSSIZE, should be JPI$_DFWSCNT.
0000    37 ;
0000    38 ;       V03-024 HWS0097         Harold Schultz              01-Aug-1984
0000    39 ;               Change max. DCL command string length from 132 to
0000    40 ;               WRK_C_INPBUFSIZ. Change max. logical name translation
0000    41 ;               size from 63 to LNM$C_NAMLENGTH.
0000    42 ;
0000    43 ;       V03-023 HWS0088         Harold Schultz              21-Jul-1984
0000    44 ;               Fix output mailbox read operation to correctly handle
0000    45 ;               an empty mailbox. Fixes problem where subprocesses
0000    46 ;               finish but never wake up parent, while running up huge
0000    47 ;               buffered I/O counts.
0000    48 ;
0000    49 ;       V03-022 HWS0064         Harold Schultz              02-May-1984
0000    50 ;               When attaching to another process, or spawning a new
0000    51 ;               new process, don't arm own attach mailbox attention
0000    52 ;               AST until just ready to hibernate. (This is to close
0000    53 ;               a window where a newly attached process tries to attach
0000    54 ;               back to the old current process before it has a chance
0000    55 ;               to marked itself detached.)
0000    56 ;               Unconceal output stream logical name.
0000    57 ;
```

```
0000     58 ;       V03-021 HWS0044         Harold Schultz          30-Mar-1984
0000     59 ;               Add SPAWN/TABLE to allow propagation of table name.
0000     60 ;               Use table search in cli name verification.
0000     61 ;               Allow ATTACH to attach to any other process in its
0000     62 ;               job, removing the the restrictions of having the input
0000     63 ;               streams be the same for both processes and that the
0000     64 ;               input device be a terminal.
0000     65 ;               When spawning a process, indicate to $CREPRC that a
0000     66 ;               CLI is being specified. (if none entered via /CLI, the
0000     67 ;               default cli is specified)
0000     68 ;               Make WRITE_AST use DCL$SPAWNOUT instead of DCL$MSGOUT.
0000     69 ;               Remove $CHIPDEF
0000     70 ;
0000     71 ;       V03-020 HWS0017         Harold Schultz          21-Feb-1984
0000     72 ;               Always signal subprocess errors - do not set inhibit bit.
0000     73 ;               Use $BRKTHRUW instead of $BRDCST.
0000     74 ;               Change /CLI specification to cli name instead of cli file spec.
0000     75 ;
0000     76 ;       V03-019 HWS0002         Harold Schultz          03-Feb-1984
0000     77 ;               Do not propagate the protection mask from a private
0000     78 ;               logical name table to a sub-process.
0000     79 ;               Fix logical name hash table processing to process
0000     80 ;               first bucket in table.
0000     81 ;
0000     82 ;       V03-018 PCG0017         Peter Georg             12-Oct-1983
0000     83 ;               Do not set final status if image count has changed.
0000     84 ;               Check for associated mbx when doing an attach.
0000     85 ;
0000     86 ;       V03-017 PCG0016         Peter George            22-Sep-1983
0000     87 ;               Fix bug in null input file processing.
0000     88 ;
0000     89 ;       V03-016 PCG0015         Peter George            15-Sep-1983
0000     90 ;               Slap colon at the end of PPD$T_INPDVI before using.
0000     91 ;
0000     92 ;       V03-015 PCG0014         Peter George            16-Aug-1983
0000     93 ;               Correctly supply address of IOSB to IO$M_TT_PROCESS QIO.
0000     94 ;               Be a little more intelligent about when to perform the QIO.
0000     95 ;
0000     96 ;       V03-014 PCG0013         Peter George            27-Jun-1983
0000     97 ;               Add /CLI qualifier.
0000     98 ;               Use event flags more consistently.
0000     99 ;               Remove code to propagate old format logical name tables.
0000    100 ;               Specify STSFLG argument to $CREPRC.
0000    101 ;               Do an IO$M_TT_PROCESS set mode QIO whenever an interactive
0000    102 ;               process comes back to life.
0000    103 ;               Remove CTRL/C and out-of-band AST warning message.
0000    104 ;
0000    105 ;       V03-013 PCG0012         Peter George            27-Jun-1983
0000    106 ;               Change MOVW in logical name code to MOVL.
0000    107 ;
0000    108 ;       V03-012 PCG0011         Peter George            15-Jun-1983
0000    109 ;               Add SPAWN/NOKEYPAD.
0000    110 ;
0000    111 ;       V03-011 PCG0010         Peter George            27-May-1983
0000    112 ;               Use DCL$FORMMSG for logical name message.
0000    113 ;
0000    114 ;       V03-010 RAS0157         Ron Schaefer            27-May-1983
```

```
0000   115 ;               Add support for new logical names to be passed thru
0000   116 ;               to the subprocess.
0000   117 ;
0000   118 ;       V03-009 PCG0009         Peter George            27-May-1983
0000   119 ;               Fix bug in SPAWN CTRL/Y processing.
0000   120 ;               Use DCL$FORMMSG.
0000   121 ;
0000   122 ;       V03-008 PCG0008         Peter George            20-Apr-1983
0000   123 ;               Add prompt string and keypad state to context that is
0000   124 ;               passed through to the subprocess.
0000   125 ;               Make spawn work in command procedures.
0000   126 ;
0000   127 ;       V03-007 PCG0005         Peter George            30-Mar-1983
0000   128 ;               Have SPAWN/NOTIFY use SPWN_T_PROCESS.
0000   129 ;               Fix bug in ATTACH_AST that allows illegal attaches.
0000   130 ;               Correctly signal process creation errors.
0000   131 ;               Process parsed input/output devices as PPF devices.
0000   132 ;
0000   133 ;       V03-006 PCG0005         Peter George            29-Mar-1983
0000   134 ;               Sort out CTX_C_KEY* symbols.
0000   135 ;
0000   136 ;       V03-005 PCG0004         Peter George            01-Mar-1983
0000   137 ;               Call DCL$GETNVAL.  Propagate keypad symbols to subprocess.
0000   138 ;
0000   139 ;       V03-004 PCG0003         Peter George            01-Feb-1983
0000   140 ;               Clean up code, fix bugs.
0000   141 ;               Add /SYMBOLS, /LOGICAL_NAMES, /NOTIFY.
0000   142 ;
0000   143 ;       V03-003 PCG0002         Peter George            14-Jan-1983
0000   144 ;               Fix SPWN deallocation problem.
0000   145 ;
0000   146 ;       V03-002 PCG0001         Peter George            18-Nov-1982
0000   147 ;               Store size of allocated TMBX structure in that structure.
0000   148 ;
0000   149 ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000   150 ;               Added $IODEF, $PSLDEF and $SSDEF.
0000   151 ;
0000   152 ;---
```

```
                                0000    154 ;
                                0000    155 ; MACRO LIBRARY CALLS
                                0000    156 ;
                                0000    157         $PRCDEF                         ;DEFINE CREPRC FLAGS
                                0000    158         PRCDEF                          ;DEFINE PROCESS WORK AREA
                                0000    159         WRKDEF                          ;DEFINE COMMAND WORK AREA
                                0000    160         PTRDEF                          ;DEFINE TOKEN DESCRIPTORS
                                0000    161         SYMDEF                          ;DEFINE SYMBOL TABLE ENTRY
                                0000    162         SPWNDEF                         ;DEFINE SPAWN LOCAL STORAGE
                                0000    163         CTXDEF                          ;DEFINE PROCESS CONTEXT MESSAGES
                                0000    164         TMBXDEF                         ;DEFINE TERMINATION MAILBOX STRUCTURE
                                0000    165         $BRKDEF                         ;DEFINE BREAKTHRU CLASSES
                                0000    166         $PPDDEF                         ;DEFINE PPD FIELDS
                                0000    167         $DIBDEF                         ;DEFINE GETDEV INFO BUFFER
                                0000    168         $JPIDEF                         ;DEFINE GETJPI ITEM CODES
                                0000    169         $DVIDEF                         ;DEFINE GETDVI ITEM CODES
                                0000    170         $SYIDEF                         ;DEFINE GETSYI ITEM CODES
                                0000    171         $PQLDEF                         ;DEFINE PROCESS QUOTA TYPE CODES
                                0000    172         $PSLDEF                         ;DEFINE PROGRAM STATUS LONGWORD FIELDS
                                0000    173         $IODEF                          ;DEFINE I/O FUNCTION CODES
                                0000    174         $SSDEF                          ;DEFINE SYSTEM STATUS CODES
                                0000    175         $LOGDEF                         ;DEFINE LOGICAL NAME ENTRY
                                0000    176         $LNMDEF                         ;DEFINE LOGICAL NAME ATTRIBUTES
                                0000    177         $LNMSTRDEF                      ;DEFINE LOGICAL NAME STRUCTURES
                                0000    178         $ACCDEF                         ;TERMINATION RECORD FORMAT
                                0000    179         $CLIMSGDEF                      ;CLI MESSAGE CODES
                                0000    180         $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS
                                0000    181         $FABDEF                         ;DEFINE FAB FIELDS
                                0000    182         $NAMDEF                         ;DEFINE NAM FIELDS
                                0000    183         $TT2DEF                         ;DEFINE DEVDEPEND2 FIELDS
                                0000    184         $DCDEF                          ;DEFINE DVI DEVICE CLASSES
                                0000    185         $$CLITABDEF                     ;DEFINE MAX PROMPT SIZE
                                0000    186
                           00000000     187         .PSECT  DCL$ZCODE,BYTE,RD,NOWRT
                                0000    188
                           00000010     0000    189 ATTMBX_MAXMSG = 16                      ; MAXIMUM SIZE OF ATTACH REQUEST MESSAGE
                                0000    190
                                0000    191 LOGINOUT:
3A 4D 45 54 53 59 53 24 53 59 53 00'    0000    192         .ASCIC  'SYS$SYSTEM:LOGINOUT'   ; IMAGE TO INITIALIZE DCL
                54 55 4F 4E 49 47 4F 4C 000C
                                   13   0000
                                        0014    193
                                        0014    194 PRCNAM_NAME:
                4C 55 21 5F 53 41 21 00' 0014    195         .ASCIC  '!AS_!UL'               ; FAO STRING FOR DEFAULT PROCESS NAME
                                   07   0014
                                        001C    196 ATTACH_NAME:
5F 48 43 41 54 54 41 24 4C 43 44 00'    001C    197         .ASCIC  'DCL$ATTACH_!XL'        ; FAO STRING FOR ATTACH LOGNAME
                                4C 58 21 0028
                                   0E   001C
                                        002B    198 OUTPUT_NAME:
5F 54 55 50 54 55 4F 24 4C 43 44 00'    002B    199         .ASCIC  'DCL$OUTPUT_!XL'        ; FAO STRING FOR OUTPUT LOGNAME
                                4C 58 21 0037
                                   0E   002B
                                        003A    200
                                        003A    201 SYS$INPUT:
      54 55 50 4E 49 24 53 59 53 00'    003A    202         .ASCIC  'SYS$INPUT'             ; DEFAULT INPUT STREAM
                                   09   003A
```

```
                                0044   203 SYS$OUTPUT:
   54 55 50 54 55 4F 24 53 59 53 00' 0044   204         .ASCIC  'SYS$OUTPUT'            ; DEFAULT OUTPUT STREAM
                              0A   0044
                                004F   205 SYS$SYSTEM:
3A 4D 45 54 53 59 53 24 53 59 53 00' 004F   206         .ASCIC  'SYS$SYSTEM:'           ; DEVICE SPEC. FOR CLI SPECIFICATION
                              0B   004F
                                005B   207 NL:
                  3A 4C 4E 5F 00' 005B   208         .ASCIC  '_NL:'                  ; NULL DEVICE
                              04   005B
                                0060   209 COM:
                  4D 4F 43 2E 00' 0060   210         .ASCIC  '.COM'                  ; DEFAULT INPUT FILE TYPE
                              04   0060
                                0065   211 LOG:
                  47 4F 4C 2E 00' 0065   212         .ASCIC  '.LOG'                  ; DEFAULT OUTPUT FILE TYPE
                              04   0065
                                006A   213
                                006A   214 CLI_NAME:                               ; KNOWN CLI NAME TABLE.
                     4C 43 44 00' 006A   215         .ASCIC  'DCL'                   ; CLI NAME FOR DCL
                              03   006A
                     52 43 4D 00' 006E   216         .ASCIC  'MCR'                   ; CLI NAME FOR MCR
                              03   006E
                  4C 4C 45 48 53 00' 0072   217         .ASCIC  'SHELL'                 ; CLI NAME FOR SHELL
                              05   0072
                              00   0078   218         .BYTE   0                       ; END OF TABLE MARKER
                                0079   219
                                0079   220 NOTIFY_MSG:
73 65 63 6F 72 70 62 75 53 07 07 00' 0079   221         .ASCIC  'Subprocess !AC has completed'
6F 63 20 73 61 68 20 43 41 21 20 73   0085
                  64 65 74 65 6C 70 6D   0091
                              1E   0079
                         0000002C  0098   222         NOTIFY_LEN = 28+15+1
```

```
                                                0098   224          .SBTTL  SPAWN COMMAND
                                                0098   225 ;+
                                                0098   226 ; DCL$SPAWN - SPAWN COMMAND
                                                0098   227 ;
                                                0098   228 ; THIS ROUTINE IS CALLED TO EXECUTE THE DCL SPAWN COMMAND.  THE SPAWN
                                                0098   229 ; COMMAND CREATES A "CLONED" SUBPROCESS WITH THE FOLLOWING CONTEXT COPIED
                                                0098   230 ; FROM THE PARENT TO THE SUBPROCESS:
                                                0098   231 ;
                                                0098   232 ;       1) ALL CLI SYMBOLS
                                                0098   233 ;       2) ALL PROCESS LOGICAL NAMES
                                                0098   234 ;       3) DEFAULT DISK AND DIRECTORY
                                                0098   235 ;       5) CURRENT PROCESS PRIVILEGES
                                                0098   236 ;       6) CURRENT COMMAND VERIFICATION STATE
                                                0098   237 ;       7) CURRENT "ON CONTROL" (OUT-OF-BAND) STATE
                                                0098   238 ;       8) CURRENT PROMPT STRING
                                                0098   239 ;       9) CURRENT KEYPAD STATE
                                                0098   240 ;
                                                0098   241 ; NO PROCESS PERMANENT OPEN FILES ARE COPIED, NOR IS ANY IMAGE OR PROCEDURE
                                                0098   242 ; CONTEXT.  THE SUBPROCESS IS SET TO PROCEDURE LEVEL 0.  LOGIN.COM IS NOT
                                                0098   243 ; EXECUTED, BOTH BECAUSE THE CONTEXT IS COPIED SEPARATELY AND TO CAUSE THE
                                                0098   244 ; SUBPROCESS TO INITIALIZE QUICKLY.  THE PARENT IS LEFT IN HIBERNATION STATE
                                                0098   245 ; UNTIL THE SUBPROCESS TERMINATES OR TRANSFERS CONTROL BACK TO THE PARENT VIA
                                                0098   246 ; THE ATTACH COMMAND.
                                                0098   247 ;
                                                0098   248 ; INPUTS:
                                                0098   249 ;
                                                0098   250 ;       R10 = ADDRESS OF COMMAND WORK AREA
                                                0098   251 ;       R11 = ADDRESS OF PROCESS WORK AREA
                                                0098   252 ;
                                                0098   253 ;
                                                0098   254 ; OUTPUTS:
                                                0098   255 ;
                                                0098   256 ;       R0 = STATUS CODE
                                                0098   257 ;
                                                0098   258 ;       R2-R9 DESTROYED.
                                                0098   259 ;-
                                                0098   260
                                                0098   261 DCL$SPAWN::
                                                0098   262 ;
                                                0098   263 ; ALLOCATE SOME SPACE FOR SPAWN STORAGE
                                                0098   264 ;
            51    00D6 8F    3C                 0098   265          MOVZWL  #SPWN_C_LENGTH,R1       ; LENGTH OF STORAGE TO ALLOCATE
                    FF60'    30                 009D   266          BSBW    DCL$ALLDYNMEM           ; ALLOCATE STORAGE
                   01 50     E8                 00A0   267          BLBS    R0,10$                  ; BRANCH IF OK
                             05                 00A3   268          RSB                             ; IF ERROR DETECTED, REPORT IT
                56   52      D0                 00A4   269 10$:     MOVL    R2,R6                   ; POINT TO SPWN STORAGE
66  00D6 8F    00   61   00   2C                 00A7   270          MOVC5   #0,(R1),#0,#SPWN_C_LENGTH,(R6)  ; ZERO THE BLOCK
                                                00AF   271                                          ; (WITHOUT DESTROYING R1)
              04 A6    51    B0                 00AF   272          MOVW    R1,SPWN_W_SIZE(R6)      ; STORE SIZE OF BLOCK
          0C A6    1065 8F    A8                 00B3   273          BISW    #SPWN_M_LOG!SPWN_M_WAIT!-       ; ASSUME /LOG, /WAIT
                                                00B9   274                  SPWN_M_CLISYM!SPWN_M_LOGNAM!-   ; COPY CLI SYMBOLS & LOGNAMES
                                                00B9   275                  SPWN_M_KEYPAD,SPWN_W_FLAGS(R6)
                A000 8F    AA                 00B9   276          BICW    #SPWN_M_CLI!SPWN_M_TABLE,-      ; ASSUME /NOCLI AND /NOTABLE
                  0C A6                        00BD   277                  SPWN_W_FLAGS(R6)
 00A3 C6    00000000'EF    B0                 00BF   278          MOVW    DCL$CRCTF,SPWN_W_PMPTCTRL(R6)   ; ASSUME /CONTROL
              0F A6    01    8E                 00C8   279          MNEGB   #1,SPWN_B_EFN(R6)       ; DO NOT SET EVENT FLAG ON TERMINATION
                                                00CC   280
```

```
                     00CC    281 ;
                     00CC    282 ; PROCESS THE VERB QUALIFIERS ON THE COMMAND LINE
                     00CC    283 ;
      FF31'   30     00CC    284 40$:    BSBW    DCL$GETDVAL                 ; GET NEXT TOKEN
    10 50     E9     00CF    285         BLBC    R0,43$                      ; IF EOL, END OF PROCESSING
 00   55     D1     00D2    286         CMPL    R5,#PTR_K_COMDQUAL          ; VERB QUALIFIER?
        0E   13     00D5    287         BEQL    42$                         ; BRANCH IF SO
 03     55   D1     00D7    288         CMPL    R5,#PTR_K_PARAMETR          ; PARAMETER (COMMAND STRING)?
        F0   12     00DA    289         BNEQ    40$                         ; IF NOT, IGNORE IT
30 A6   51   7D     00DC    290         MOVQ    R1,SPWN_Q_CMDSTR(R6)        ; SAVE DESCRIPTOR OF COMMAND STRING
        EA   11     00E0    291         BRB     40$
     0171    31     00E2    292 43$:    BRW     80$                         ; END OF COMMAND PARSING
      FF18'  30     00E5    293 42$:    BSBW    DCL$GETNVAL                 ; GET QUALIFIER NUMBER
00'8F   51   91     00E8    294         CMPB    R1,#CLI$K_SPAW_WAIT         ; /WAIT?
        27   13     00EC    295         BEQL    49$
00'8F   51   91     00EE    296         CMPB    R1,#CLI$K_SPAW_LOG          ; /LOG?
        2F   13     00F2    297         BEQL    47$
00'8F   51   91     00F4    298         CMPB    R1,#CLI$K_SPAW_SYMB         ; /SYMBOLS?
        37   13     00F8    299         BEQL    50$
00'8F   51   91     00FA    300         CMPB    R1,#CLI$K_SPAW_LOGI         ; /LOGICAL_NAMES?
        3F   13     00FE    301         BEQL    51$
00'8F   51   91     0100    302         CMPB    R1,#CLI$K_SPAW_NOTI         ; /NOTIFY?
        49   13     0104    303         BEQL    52$
00'8F   51   91     0106    304         CMPB    R1,#CLI$K_SPAW_CARR         ; /CARRIAGE_CONTROL?
        53   13     010A    305         BEQL    531$
00'8F   51   91     010C    306         CMPB    R1,#CLI$K_SPAW_KEYP         ; /KEYPAD?
        60   13     0110    307         BEQL    532$
     006D    31     0112    308         BRW     60$                         ; CHECK OTHER QUALIFIER POSSIBILITIES
                     0115    309
                     0115    310 49$:    SETBIT  SPWN_V_WAIT,SPWN_W_FLAGS(R6)    ; ASSUME /WAIT
 AF 53   00   E1     0119    311         BBC     #PTR_V_NEGATE-PTR_V_FLAGS,R3,40$ ; IGNORE IF NOT /NOWAIT
                     011D    312         CLRBIT  SPWN_V_WAIT,SPWN_Q_FLAGS(R6)    ; CLEAR WAIT FLAG
            A9   11  0121    313         BRB     40$
                     0123    314 47$:    SETBIT  SPWN_V_LOG,SPWN_W_FLAGS(R6)     ; ASSUME /LOG
 A1 53   00   E1     0127    315         BBC     #PTR_V_NEGATE-PTR_V_FLAGS,R3,40$ ; IGNORE IF NOT /NOLOG
                     012B    316         CLRBIT  SPWN_V_LOG,SPWN_W_FLAGS(R6)     ; CLEAR LOG MESSAGE FLAG
            9B   11  012F    317         BRB     40$
                     0131    318 50$:    SETBIT  SPWN_V_CLISYM,SPWN_W_FLAGS(R6)  ; ASSUME /SYMBOLS
 93 53   00   E1     0135    319         BBC     #PTR_V_NEGATE-PTR_V_FLAGS,R3,40$ ; IGNORE IF NOT /NOSYMBOLS
                     0139    320         CLRBIT  SPWN_V_CLISYM,SPWN_Q_FLAGS(R6)  ; CLEAR CLISYM FLAG
            8D   11  013D    321         BRB     40$
                     013F    322 51$:    SETBIT  SPWN_V_LOGNAM,SPWN_W_FLAGS(R6)  ; ASSUME /LOGICAL_NAMES
 5E 53   00   E1     0144    323         BBC     #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ ; IGNORE IF NOT /NOLOGICAL_NAMES
                     0148    324         CLRBIT  SPWN_V_LOGNAM,SPWN_Q_FLAGS(R6)  ; CLEAR LOGNAM FLAG
            57   11  014D    325         BRB     41$
                     014F    326 52$:    SETBIT  SPWN_V_NOTIFY,SPWN_W_FLAGS(R6)  ; ASSUME /NOTIFY
 4E 53   00   E1     0154    327         BBC     #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ ; IGNORE IF NOT /NONOTIFY
                     0158    328         CLRBIT  SPWN_V_NOTIFY,SPWN_Q_FLAGS(R6)  ; CLEAR NOTIFY FLAG
            47   11  015D    329         BRB     41$
00A3 C6 00000000'EF  B0  015F    330 531$:   MOVW    DCL$CRLF,SPWN_W_PMPTCTRL(R6)    ; ASSUME /CONTROL
 3A 53   00   E1     0168    331         BBC     #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ ; IGNORE IF NOT /NOCONTROL
     00A3 C6  B4     016C    332         CLRW    SPWN_W_PMPTCTRL(R6)             ; SET /NOCONTROL
            34   11  0170    333         BRB     41$
                     0172    334 532$:   SETBIT  SPWN_V_KEYPAD,SPWN_W_FLAGS(R6)  ; ASSUME /KEYPAD
 2B 53   00   E1     0177    335         BBC     #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ ; IGNORE IF NOT /NOKEYPAD
                     017B    336         CLRBIT  SPWN_V_KEYPAD,SPWN_Q_FLAGS(R6)  ; CLEAR KEYPAD FLAG
            24   11  0180    337         BRB     41$
```

```
                     0182   338
00'8F   51   91      0182   339 60$:    CMPB    R1,#CLI$K_SPAW_INPU     ; /INPUT?
        21   13      0186   340         BEQL    45$
00'8F   51   91      0188   341         CMPB    R1,#CLI$K_SPAW_OUTP     ; /OUTPUT?
        33   13      018C   342         BEQL    46$
00'8F   51   91      018E   343         CMPB    R1,#CLI$K_SPAW_PROC     ; /PROCESS_NAME?
        45   13      0192   344         BEQL    48$
00'8F   51   91      0194   345         CMPB    R1,#CLI$K_SPAW_CLI      ; /CLI?
        51   13      0198   346         BEQL    481$
00'8F   51   91      019A   347         CMPB    R1,#CLI$K_SPAW_TABL     ; /TABLE?
        63   13      019E   348         BEQL    485$
00'8F   51   91      01A0   349         CMPB    R1,#CLI$K_SPAW_PROM     ; /PROMPT?
        75   13      01A4   350         BEQL    53$
      FF23   31      01A6   351 41$:    BRW     40$                     ; IF NONE OF THE ABOVE QUALS, IGNORE IT
                     01A9   352
                     01A9   353 45$:    CLRBIT  SPWN_V_INPUT,SPWN_W_FLAGS(R6)   ; ASSUME NO VALUE PRESENT
    02   54   91      01AE   354         CMPB    R4,#PTR_K_COLON                 ; IS A VALUE PRESENT?
         F3   12      01B1   355         BNEQ    41$                             ; IF NOT, USE DEFAULT
                     01B3   356         SETBIT  SPWN_V_INPUT,SPWN_W_FLAGS(R6)   ; INDICATE VALUE PRESENT
    FE45'    30      01B8   357         BSBW    DCL$GETDVAL                     ; GET /INPUT VALUE
20 A6   51   7D      01BB   358         MOVQ    R1,SPWN_Q_INPUT(R6)             ; USE EXPLICIT INPUT STREAM
        E5   11      01BF   359         BRB     41$
                     01C1   360 46$:    CLRBIT  SPWN_V_OUTPUT,SPWN_W_FLAGS(R6)  ; ASSUME NO VALUE PRESENT
    02   54   91      01C6   361         CMPB    R4,#PTR_K_COLON                 ; IS A VALUE PRESENT?
         DB   12      01C9   362         BNEQ    41$                             ; IF NOT, USE DEFAULT
                     01CB   363         SETBIT  SPWN_V_OUTPUT,SPWN_W_FLAGS(R6)  ; INDICATE VALUE PRESENT
    FE2D'    30      01D0   364         BSBW    DCL$GETDVAL                     ; GET /OUTPUT VALUE
28 A6   51   7D      01D3   365         MOVQ    R1,SPWN_Q_OUTPUT(R6)            ; USE EXPLICIT OUTPUT STREAM
        CD   11      01D7   366         BRB     41$
    02   54   91      01D9   367 48$:    CMPB    R4,#PTR_K_COLON                 ; IS A VALUE PRESENT?
         C8   12      01DC   368         BNEQ    41$                             ; IF NOT, USE DEFAULT
    FE1F'    30      01DE   369         BSBW    DCL$GETDVAL                     ; GET VALUE
18 A6   51   7D      01E1   370         MOVQ    R1,SPWN_Q_PRCNAM(R6)            ; USE EXPLICIT PROCESS NAME
                     01E5   371         SETBIT  SPWN_V_PRCNAM,SPWN_W_FLAGS(R6)  ; SET PROCESS NAME SPECIFIED
        BB   12      01E9   372         BNEQ    41$
                     01EB   373 481$:   CLRBIT  SPWN_V_CLI,SPWN_W_FLAGS(R6)     ; ASSUME /NOCLI
B2 53   00   E0      01F0   374         BBS     #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ ; FLAG OK AS IS IF /NOCLI
                     01F4   375         SETBIT  SPWN_V_CLI,SPWN_W_FLAGS(R6)     ; SET /CLI PRESENT FLAG
    FE04'    30      01F9   376         BSBW    DCL$GETDVAL                     ; GET /CLI VALUE
00C6 C6   51   7D      01FC   377         MOVQ    R1,SPWN_Q_CLI(R6)               ; SAVE DESCRIPTOR OF CLI NAME
        A3   11      0201   378         BRB     41$
                     0203   379 485$:   CLRBIT  SPWN_V_TABLE,SPWN_W_FLAGS(R6)   ; ASSUME /NOTABLE
9A 53   00   E0      0208   380         BBS     #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ ; FLAG OK AS IS IF /NOTABLE
                     020C   381         SETBIT  SPWN_V_TABLE,SPWN_W_FLAGS(R6)   ; SET /TABLE PRESENT FLAG
    FDEC'    30      0211   382         BSBW    DCL$GETDVAL                     ; GET /TABLE VALUE
00CE C6   51   7D      0214   383         MOVQ    R1,SPWN_Q_TABLE(R6)             ; SAVE DESCRIPTOR OF TABLE NAME
        8B   11      0219   384         BRB     41$
                     021B   385 53$:    CLRBIT  SPWN_V_PROMPT,SPWN_W_FLAGS(R6)  ; ASSUME /NOPROMPT
30 53   E8      0220   386         BLBS    R3,533$                         ; BRANCH IF SO
                     0223   387         SETBIT  SPWN_V_PROMPT,SPWN_W_FLAGS(R6)  ; SET /PROMPT
00'8F   90      0228   388         MOVB    #DCL$C_PROMPTLEN,-              ; ASSUME NO VALUE SPECIFIED
00A2 C6                022B   389                 SPWN_B_PROMPTLEN(R6)            ;
    54   DD      022E   390         PUSHL   R4                              ; SAVE THE TERMINATOR
0000'8F   28      0230   391         MOVC3   #DCL$C_PROMPTLEN,DCL$CRLF,-     ; GET PROMPT STRING
00A3 C6                0239   392                 SPWN_W_PMPTCTRL(R6)             ;
    54 8ED0      023C   393         POPL    R4                              ; RESTORE THE TERMINATOR
02   54   91      023F   394         CMPB    R4,#PTR_K_COLON                 ; IS A VALUE PRESENT?
```

```
                     0F   12  0242   395          BNEQ    533$                                    ; USE DEFAULT PROMPT
                   FDB9'  30  0244   396          BSBW    DCL$GETDVAL                             ; GET VALUE
00A2 C6          51   03   81  0247   397          ADDB3   #3,R1,SPWN_B_PROMPTLEN(R6)              ; GET PROMPT LENGTH
                 62   20   28  024D   398          MOVC3   #ENT_K_MAX_PROMPT,(R2),-                ; GET PROMPT STRING
                 00A6 C6         0250   399                  SPWN_G_PROMPT(R6)                   ;
                   FE76   31     0253   400 533$:   BRW     40$
                                  0256   401
                                  0256   402 ;
                                  0256   403 ; THE PARSING IS COMPLETE, AND ALL OPTIONS ARE SET INTO THE SPWN
                                  0256   404 ; BLOCK.  NOW PERFORM THE SPAWN OPERATION.
                                  0256   405 ;
                     07   10     0256   406 80$:    BSBB    DCL$SPAWN2                              ; PERFORM THE SPAWN OPERATION
                  03 50   E9     0258   407          BLBC    R0, 90$                                 ; TEST FOR FAILURE OF SPAWN ITSELF
                50   51   D0     025B   408          MOVL    R1, R0                                  ; GET SUBPROCESS TERMINATION STATUS
                          05     025E   409 90$:    RSB
```

```
                     025F   411          .SBTTL  SPAWN A SUBPROCESS
                     025F   412 ;---
                     025F   413 ;
                     025F   414 ; THIS ROUTINE IS CALLED TO PERFORM THE ACTUAL CREATION OF THE SUBPROCESS.
                     025F   415 ;
                     025F   416 ; INPUTS:
                     025F   417 ;
                     025F   418 ;       R6 = ADDRESS OF SPWN PARAMETER BLOCK
                     025F   419 ;       R11 = ADDRESS OF PRC AREA
                     025F   420 ;
                     025F   421 ; OUTPUTS:
                     025F   422 ;
                     025F   423 ;       R0 = STATUS
                     025F   424 ;       R1 = FINAL STATUS FROM COMPLETED SUBPROCESS
                     025F   425 ;       SPWN_L_PID(R6) = PID OF SUBPROCESS (IF NOWAIT FLAG SPECIFIED)
                     025F   426 ;---
                     025F   427
                     025F   428 DCL$SPAWN2::
01BC 8F     BB       025F   429          PUSHR   #^M<R2,R3,R4,R5,R7,R8>          ; SAVE REGISTERS
                     0263   430
                     0263   431 ;
                     0263   432 ; CONTROL/Y AST'S ARE DISABLED THROUGHOUT THIS COMMAND, TO ENSURE THAT
                     0263   433 ; MAILBOXES WHICH ARE CREATED HERE ARE CORRECTLY DELETED, ETC.  AS A
                     0263   434 ; RESULT, WE MUST PERIODICALLY CHECK THE HANGUP FLAG IN CASE A HANGUP
                     0263   435 ; IS DETECTED WHILE WE ARE OPERATING.
                     0263   436 ;
00B4 CB     D0       0263   437          MOVL    PRC_L_OUTOFBAND(R11),-         ; SAVE OUT-OF-BAND ENABLE MASK
   58 A6              0267   438                  SPWN_L_OUTOFBAND(R6)           ;
      51    D4        0269   439          CLRL    R1                             ; DISABLE ALL OUT-OF-BAND AST'S
    FD92'   30        026B   440          BSBW    DCL$RESETOOB                   ;
                     026E   441
                     026E   442 ;
                     026E   443 ; PRE-CLEAR THE TERMINATION EVENT FLAG, IF ONE IS SPECIFIED.
                     026E   444 ;
        0F A6   95   026E   445          TSTB    SPWN_B_EFN(R6)                 ; EVENT FLAG SPECIFIED?
           0A   19   0271   446          BLSS    2$                             ; BRANCH IF NONE
                     0273   447          $CLREF_S EFN=SPWN_B_EFN(R6)            ; PRE-CLEAR IT
                     027D   448
                     027D   449 ;
                     027D   450 ; SAVE THE PREVIOUS ACCESS MODE, FOR PROBING RETURN STATUS AND DELIVERING
                     027D   451 ; TERMINATION ASTS.
                     027D   452 ;
           50   DC   027D   453 2$:      MOVPSL  R0                             ; GET CURRENT PSL
        02 16   EF   027F   454          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,-  ; EXTRACT PREVIOUS ACMODE
        50 50        0282   455                  R0,R0                          ;
     0E A6 50   90   0284   456          MOVB    R0,SPWN_B_ACMODE(R6)           ; SAVE THE ACCESS MODE
                     0288   457
                     0288   458 ;
                     0288   459 ; ALLOCATE SOME SCRATCH STORAGE ON THE STACK (1 - 1/2 PAGES)
                     0288   460 ;
   5E   FD00 CE   9E 0288   461          MOVAB   -768(SP),SP                    ; ALLOCATE SCRATCH BUFFER ON STACK
             5E   DD 028D   462          PUSHL   SP                             ; CONSTRUCT DESCRIPTOR OF BUFFER
   7E   0300 8F   3C 028F   463          MOVZWL  #768,-(SP)                     ;
        58   5E   D0 0294   464          MOVL    SP,R8                          ; AND POINT TO IT
                     0297   465
                     0297   466 ;
                     0297   467 ; FIND A TERMINATION MAILBOX SLOT AND INITIALIZE IT FOR THIS SUBPROCESS.
```

```
                        0297   468 ;
                        0297   469         ASSUME  TMBX_L_LINK EQ 0                      ;
    57   74 AB    9E     0297   470         MOVAB   PRC_C_TMBX(R11),R7                   ; GET ADDRESS OF TMBX BLOCK LIST HEA
         57 67    D0     029B   471 10$:     MOVL    TMBX_C_LINK(R7),R7                   ; GET NEXT BLOCK IN LIST
            09    12     029E   472         BNEQ    20$                                  ; BRANCH IF NOT END OF LIST
              0ACE 30     02A0   473         BSBW    CREATE_TMBX                          ; CREATE NEW MBX IF END OF LIST
            03 50 E8     02A3   474         BLBS    R0,20$                               ; CONTINUE IF NO ERROR
              00FA 31     02A6   475         BRW     230$                                 ; EXIT IF ERROR
    04      08 A7 91     02A9   476 20$:     CMPB    TMBX_B_REFS(R7),#TMBX_C_MAXREFS      ; DOES MBX HAVE ANY OPEN SLOTS?
            EC    13     02AD   477         BEQL    10$                                  ; NO, LOOK AT NEXT MBX IN LIST
            08 A7 96     02AF   478         INCB    TMBX_B_REFS(R7)                      ; YES, GRAB ONE, INCR REFERENCE COUN
08 A6       04 A7 B0     02B2   479         MOVW    TMBX_W_UNIT(R7),SPWN_W_UNIT(R6)      ; COPY THE UNIT NUMBER
                        02B7   480 ;
                        02B7   481 ; CHECK FOR KNOWN CLI'S  TO WHICH TO PASS CONTEXT IF /CLI
                        02B7   482 ; WAS SPECIFIED.
                        02B7   483 ;
        7E   58 7D      02B7   484         MOVQ    R8,-(SP)                             ; SAVE WORK REGISTERS
2F 0C A6     0D E1      02BA   485         BBC     #SPWN_V_CLI,SPWN_W_FLAGS(R6),50$     ; BRANCH IF NO CLI WAS SPECIFED
 52   00C6 C6 7D      02BF   486         MOVQ    SPWN_Q_CLI(R6),R2                    ; GET CLI DESCRIPTOR
 58   FDA2 CF 9E      02C4   487         MOVAB   CLI_NAME,R8                          ; GET TABLE ADDR. OF KNOWN CLI'S
                        02C9   488
        59   68 9A      02C9   489 21$:     MOVZBL  (R8),R9                              ; EXTRACT LENGTH
             16 13      02CC   490         BEQL    40$                                  ; IF EQ, SPECIFIED CLI NOT KNOWN
                        02CE   491
        7E   52 7D      02CE   492         MOVQ    R2,-(SP)                             ; SAVE DESCRIPTOR
 00     63   52 2D      02D1   493         CMPC5   R2,(R3),#0,-                         ; COMPARE TO THIS TABLE ENTRY
   01 A8      59        02D5   494                 R9,1(R8)
             11 13      02D8   495         BEQL    45$                                  ; IF EQ, THIS CLI IS KNOWN
                        02DA   496
        52   8E 7D      02DA   497         MOVQ    (SP)+,R2                             ; RESTORE DESCRIPTOR
 58     01 A849 9E      02DD   498         MOVAB   1(R8)[R9],R8                         ; SET UP TO CHECK NEXT ENTRY
             E5 11      02E2   499         BRB     21$                                  ; CHECK NEXT TABLE ENTRY
                        02E4   500
                        02E4   501 40$:     SETBIT  SPWN_V_NOCTX,SPWN_W_FLAGS(R6)        ; DON'T SEND CONTEXT TO FOR THIS CLI
             03 11      02E9   502         BRB     50$
        52   8E 7D      02EB   503 45$:     MOVQ    (SP)+,R2                             ; RESTORE STACK
                        02EE   504 ;
                        02EE   505 ; PROPAGATE EITHER THE SPECIFIED CLI AND TABLE NAMES OR THE DEFAULT NAMES TO
                        02EE   506 ; THE SPAWN CLI AND TABLE SPECIFICATION TABLES.
                        02EE   507 ;
 58     00C6 C6 3C      02EE   508 50$:     MOVZWL  SPWN_Q_CLI(R6),R8                    ; GET LENGTH OF CLI NAME, IF SPEC.
 59     00CE C6 3C      02F3   509         MOVZWL  SPWN_Q_TABLE(R6),R9                  ; GET LENGTH OF TABLE NAME, IF SPECI
40 0C A6     0F E1      02F8   510         BBC     #SPWN_V_TABLE,SPWN_W_FLAGS(R6),218$  ; SKIP IF NO TABLE SPECIFIED
24 0C A6     0D E1      02FD   511         BBC     #SPWN_V_CLI,SPWN_W_FLAGS(R6),216$    ; SKIP IF NO CLI SPECIFIED
                        0302   512 ;
                        0302   513 ; BOTH CLI AND NAME WERE SPECIFIED
                        0302   514 ;
00000000'GF     58 90   0302   515         MOVB    R8,G^CTL$GT_SPAWNCLI                 ; SET SIZE FIELD
   00CA D6      58 28   0309   516         MOVC    R8,@SPWN_Q_CLI+4(R6),-               ; MOVE CLI SPECIFICATION
     00000001'GF        030E   517                 G^CTL$GT_SPAWNCLI+1
00000000'GF     59 90   0313   518 214$:    MOVB    R9,G^CTL$GT_SPAWNTABLE               ; SET LENGTH OF TABLE NAME
   00D2 D6      59 28   031A   519         MOVC    R9,@SPWN_Q_TABLE+4(R6),-             ; COPY TABLE NAME
     00000001'GF        031F   520                 G^CTL$GT_SPAWNTABLE+1
             5F 11      0324   521         BRB     228$
                        0326   522 ;
                        0326   523 ; JUST TABLE NAME SPECIFIED
                        0326   524 ;
```

```
                58   00000000'GF   9A  0326    525 216$:   MOVZBL  G^CTL$GT_CLINAME,R8                  ; GET LENGTH OF CLI NAME
                                   58  D6  032D    526         INCL    R8                              ; INCLUDE LENGTH BYTE
                00000000'GF    58  28  032F    527         MOVC    R8,G^CTL$GT_CLINAME,-           ; GET DEFAULT CLI NAME
                     00000000'GF       0336    528                 G^CTL$GT_SPAWNCLI
                                   D6  11  033B    529         BRB     214$                            ; NOW TRANSER TABLE NAME
                                       033D    530 ;
                                       033D    531 ;       NO TABLE NAME SPECIFIED. CHECK IF CLI NAME SPECIFIED
                                       033D    532 ;
                   19 0C A6    0D  E1  033D    533 218$:   BBC     #SPWN_V_CLI,SPWN_W_FLAGS(R6),220$  ; SKIP IF CLI NOT SPECIFIED
                     00000000'GF   94  0342    534         CLRB    G^CTL$GT_SPAWNTABLE             ; ZERO TABLE SPECIFICATION
                00000000'GF    58  90  0348    535         MOVB    R8,G^CTL$GT_SPAWNCLI            ; SET SIZE FIELD
                     00CA D6    58  28  034F    536         MOVC    R8,@SPWN_Q_CLI+4(R6),-          ; MOVE CLI SPECIFICATION
                     00000001'GF       0354    537                 G^CTL$GT_SPAWNCLI+1
                                   2A  11  0359    538         BRB     228$
                                       035B    539 ;
                                       035B    540 ;       NEITHER CLI OR TABLE SPECIFIED. JUST COPY DEFAULTS
                                       035B    541 ;
                58   00000000'GF   9A  035B    542 220$:   MOVZBL  G^CTL$GT_CLINAME,R8                  ; GET LENGTH OF CLI NAME
                                   58  D6  0362    543         INCL    R8                              ; INCLUDE LENGTH BYTE
                00000000'GF    58  28  0364    544         MOVC    R8,G^CTL$GT_CLINAME,-           ; COPY CLI NAME
                     00000000'GF       036B    545                 G^CTL$GT_SPAWNCLI
                58   00000000'GF   9A  0370    546         MOVZBL  G^CTL$GT_TABLENAME,R8           ; GET LENGTH OF TABLE NAME
                                   58  D6  0377    547         INCL    R8                              ; INCLUDE LENGTH BYTE
                00000000'GF    58  28  0379    548         MOVC    R8,G^CTL$GT_TABLENAME,-         ; COPY TABLE NAME
                     00000000'GF       0380    549                 G^CTL$GT_SPAWNTABLE
                        58      8E   7D  0385    550 228$:   MOVQ    (SP)+,R8                        ; RESTORE WORK REG.
                                       0388    551 ;
                                       0388    552 ; CREATE A COMMUNICATIONS MAILBOX TO COPY THE CONTEXT TO THE SUBPROCESS
                                       0388    553 ;
                5F 0C A6    0E  E0  0388    554         BBS     #SPWN_V_NOCTX,SPWN_W_FLAGS(R6),22$; BRANCH IF UNKNOWN CLI WAS SPECIF
                                       038D    555         $CREMBX_S CHAN=SPWN_W_CHAN(R6),-        ; CREATE COMMUNICATIONS MAILBOX
                                       038D    556                 MAXMSG=#CTX_C_MAXLEN            ;
                        4C 50   E9  03A3    557 230$:   BLBC    R0,95$                          ; BRANCH ON ERROR
                                       03A6    558
                                       03A6    559 ;
                                       03A6    560 ; GET THE NAME OF THAT MAILBOX USING $GETDVI.
                                       03A6    561 ;
                           7E   D4  03A6    562         CLRL    -(SP)                           ; CREATE GETDVI ITEM LIST
                              10 A6    9F  03A8    563         PUSHAB  SPWN_Q_MBXNAM(R6)               ; ADDRESS OF WORD TO RECEIVE LENGTH
                        7E      68   7D  03AB    564         MOVQ    (R8),-(SP)                      ; SET DESCRIPTOR OF BUFFER
                     02 AE      20   B0  03AE    565         MOVW    #DVI$_DEVNAM,2(SP)              ; REQUEST COMPLETE DEVICE NAME
                        50      5E   D0  03B2    566         MOVL    SP,R0                           ;
                                       03B5    567         $GETDVIW_S  CHAN=SPWN_W_CHAN(R6),-      ; GET MAILBOX DEVICE NAME
                                       03B5    568                 IOSB=SPWN_Q_IOSB(R6),-          ;
                                       03B5    569                 EFN=#EXE$C_SYSEFN,-             ;
                                       03B5    570                 ITMLST=(R0)                     ;
                           5E   10   C0  03D1    571         ADDL    #4*4,SP                         ; POP GETDVI ITEM LIST
                              1B 50    E9  03D4    572         BLBC    R0,95$                          ; BRANCH IF ERROR DETECTED
                        50      38 A6    3C  03D7    573         MOVZWL  SPWN_Q_IOSB(R6),R0              ; GET IOSB STATUS
                              14 50    E9  03DB    574         BLBC    R0,95$                          ; BRANCH IF ERROR DETECTED
                                       03DE    575
                                       03DE    576 ;
                                       03DE    577 ; SAVE THE NAME OF THE COMMUNICATIONS MAILBOX IN THE SPWN BLOCK.
                                       03DE    578 ; DEALLOCATE SPACE FOR THE MAILBOX NAME FROM THE SCRATCH STORAGE.
                                       03DE    579 ;
                14 A6    04 A8   D0  03DE    580         MOVL    4(R8),SPWN_Q_MBXNAM+4(R6)       ; SET ADDRESS OF BUFFER
                   68    10 A6   A2  03E3    581         SUBW    SPWN_Q_MBXNAM(R6),(R8)          ; MARK DEVICE NAME NO LONGER SCRATCH
```

```
                04 A8   10 A6    C0  03E7   582          ADDL    SPWN_Q_MBXNAM(R6),4(R8)         ;
                                   03EC   583
                                   03EC   584 ;
                                   03EC   585 ; CREATE AN ATTACH MAILBOX TO HANDLE RE-ATTACH REQUESTS TO THIS PROCESS.
                                   03EC   586 ;
                     08F3   30     03EC   587 22$:    BSBW    CREATE_ATTMBX                   ; CREATE OUR ATTACH MAILBOX
                   03 50    E8     03EF   588         BLBS    R0,25$                          ; BRANCH IF SUCCESS
                    02CD    31     03F2   589 95$:    BRW     SPAWN_EXIT                      ; BRANCH IF ERROR DETECTED
                                   03F5   590
                                   03F5   591 ;
                                   03F5   592 ; IF /NOTIFY WAS SPECIFIED, CHECK THAT IT IS ALLOWED.
                                   03F5   593 ;
                       08    E1    03F5   594 25$:    BBC     #SPWN_V_NOTIFY,-                ; SKIP IF /NONOTIFY
                  11 0C A6         03F7   595                 SPWN_Q_FLAGS(R6),30$            ;
                00038250 8F    D0  03FA   596         MOVL    #CLI$_NOTIFY,R0                 ; ASSUME NOTIFY NOT ALLOWED
              EC 68 AB     06    E0  0401   597         BBS     #PRC_V_MODE,PRC_W_FLAGS(R11),95$; NOT ALLOWED IN BATCH JOBS
                           02    E0  0406   598         BBS     #SPWN_V_WAIT,-                  ; NOT ALLOWED IN /WAIT JOBS
                  E7 0C A6         0408   599                 SPWN_Q_FLAGS(R6),95$            ;
                                   040B   600
                                   040B   601 ;
                                   040B   602 ; CHECK THAT COMMAND STRING IS NOT TOO LONG.
                                   040B   603 ;
              50 00038018 8F    D0  040B   604 30$:    MOVL    #CLI$_BUFOVF,R0                 ; ASSUME COMMAND BUFFER OVERFLOW
               0100 8F    30 A6   B1  0412   605         CMPW    SPWN_Q_CMDSTR(R6),#WRK_C_INPBUFSIZ  ; COMMAND STRING TOO LONG?
                           D8    1A  0418   606         BGTRU   95$                             ; BRANCH IF SO
                                   041A   607
                                   041A   608 ;
                                   041A   609 ; CHECK THAT PROMPT STRING IS NOT TOO LONG.
                                   041A   610 ;
           50 000388FA 8F    D0  041A   611         MOVL    #CLI$_STRTOOLNG,R0              ; ASSUME PROMPT IS TOO LONG
                 00A2 C6    91  0421   612         CMPB    SPWN_B_PROMPTLEN(R6),-          ; PROMPT STRING TOO LONG?
                        23         0425   613                 #ENT_K_MAX_PROMPT+3             ;
                        CA    1A   0426   614         BGTRU   95$                             ; BRANCH IF SO
                                   0428   615
                                   0428   616 ;
                                   0428   617 ; PROCESS THE INPUT AND OUTPUT STREAMS.
                                   0428   618 ;
                     02E5    30    0428   619         BSBW    VERIFY_INPUT                    ; VERIFY INPUT STREAM
                   C4 50     E9    042B   620         BLBC    R0,95$                          ; BRANCH IF ERROR
                     04BE    30    042E   621         BSBW    VERIFY_OUTPUT                   ; VERIFY OUTPUT STREAM
                   BE 50     E9    0431   622         BLBC    R0,95$                          ; BRANCH IF ERROR
                                   0434   623
                                   0434   624 ;
                                   0434   625 ; DETERMINE BASE PRIORITY FOR NEW SUBPROCESS FROM CURRENT BASE PRIORITY
                                   0434   626 ;
                        7E    7C   0434   627         CLRQ    -(SP)                           ; CREATE GETJPI ITEM LIST
                     48 A6    DF   0436   628         PUSHAL  SPWN_L_PRIB(R6)                 ; SET BUFFER ADDRESS
              03090004 8F    DD   0439   629         PUSHL   #JPI$_PRIB@16+4                 ; REQUEST CURRENT BASE PRIORITY
                                   043F   630
                                   043F   631 ;
                                   043F   632 ; DETERMINE CURRENT IMAGE COUNT (FOR USE BY TERMINATION AST)
                                   043F   633 ;
                        7E    D4   043F   634         CLRL    -(SP)                           ; NO RETLEN ADDRESS
                     5C A6    DF   0441   635         PUSHAL  SPWN_L_IMAGCNT(R6)              ; SET BUFFER ADDRESS
              041A0004 8F    DD   0444   636         PUSHL   #JPI$_IMAGECOUNT@16+4           ; REQUEST CURRENT IMAGE COUNT
                                   044A   637
                                   044A   638 ;
```

```
                     044A    639 ; DETERMINE CURRENT VALUES OF NON-DEDUCTIBLE QUOTAS, AND CREATE A QUOTA
                     044A    640 ; LIST WHICH WILL BE USED TO SET THE QUOTAS OF THE SUBPROCESS.
                     044A    641 ;
 53   60 A6   9E     044A    642         MOVAB   SPWN_G_QUOTAS(R6),R3            ; GET ADDRESS OF SPACE FOR LIST
       7E   D4     044E    643         CLRL    -(SP)                           ; NO RETLEN ADDRESS
    83   01   90     0450    644         MOVB    #PQL$_ASTLM,(R3)+               ; SET TYPE CODE IN PQL LIST
       83   DF     0453    645         PUSHAL  (R3)+                           ; SET BUFFER ADDRESS
04090004   8F   DD     0455    646         PUSHL   #JPI$_ASTLM@16+4                ; REQUEST CURRENT ASTLM
       7E   D4     045B    647         CLRL    -(SP)                           ; NO RETLEN ADDRESS
    83   02   90     045D    648         MOVB    #PQL$_BIOLM,(R3)+               ; SET TYPE CODE IN PQL LIST
       83   DF     0460    649         PUSHAL  (R3)+                           ; SET BUFFER ADDRESS
03100004   8F   DD     0462    650         PUSHL   #JPI$_BIOLM@16+4                ; REQUEST CURRENT BIOLM
       7E   D4     0468    651         CLRL    -(SP)                           ; NO RETLEN ADDRESS
    83   05   90     046A    652         MOVB    #PQL$_DIOLM,(R3)+               ; SET TYPE CODE IN PQL LIST
       83   DF     046D    653         PUSHAL  (R3)+                           ; SET BUFFER ADDRESS
03130004   8F   DD     046F    654         PUSHL   #JPI$_DIOLM@16+4                ; REQUEST CURRENT DIOLM
       7E   D4     0475    655         CLRL    -(SP)                           ; NO RETLEN ADDRESS
    83   0B   90     0477    656         MOVB    #PQL$_WSDEFAULT,(R3)+           ; SET TYPE CODE IN PQL LIST
       83   DF     047A    657         PUSHAL  (R3)+                           ; SET BUFFER ADDRESS
04030004   8F   DD     047C    658         PUSHL   #JPI$_DFWSCNT@16+4              ; REQUEST CURRENT WSDEFAULT
       7E   D4     0482    659         CLRL    -(SP)                           ; NO RETLEN ADDRESS
    83   0A   90     0484    660         MOVB    #PQL$_WSQUOTA,(R3)+             ; SET TYPE CODE IN PQL LIST
       83   DF     0487    661         PUSHAL  (R3)+                           ; SET BUFFER ADDRESS
04020004   8F   DD     0489    662         PUSHL   #JPI$_WSQUOTA@16+4              ; REQUEST CURRENT WSQUOTA
       7E   D4     048F    663         CLRL    -(SP)                           ; NO RETLEN ADDRESS
    83   0D   90     0491    664         MOVB    #PQL$_WSEXTENT,(R3)+            ; SET TYPE CODE IN PQL LIST
       83   DF     0494    665         PUSHAL  (R3)+                           ; SET BUFFER ADDRESS
04160004   8F   DD     0496    666         PUSHL   #JPI$_WSEXTENT@16+4             ; REQUEST CURRENT WSEXTENT
    83   00   90     049C    667         MOVB    #PQL$_LISTEND,(R3)+             ; TERMINATE PQL LIST
    50   5E   D0     049F    668         MOVL    SP,R0                           ; GET ADDRESS OF ITEM LIST
                     04A2    669         $GETJPIW_S  ITMLST=(R0),-               ; GET CURRENT QUOTA VALUES
                     04A2    670                 IOSB=SPWN_Q_IOSB(R6),-          ;
                     04A2    671                 EFN=#EXE$C_SYSEFN               ;
5E   00000064   8F   C0     04BA    672         ADDL    #8*12+4,SP              ; POP ITEM LIST
                     04C1    673
                     04C1    674 ;
                     04C1    675 ; INSERT THE CURRENT SPWN BLOCK INTO THE LINKED LIST OF PENDING SUBPROCESSES,
                     04C1    676 ; SO THAT IN CASE THE SUBPROCESS DIES WHILE WE ARE ATTEMPTING TO FEED IT
                     04C1    677 ; CONTEXT, THE SUBPROCESS TERMINATION AST ROUTINE CAN DO SOMETHING ABOUT IT.
                     04C1    678 ;
                     04C1    679         SETBIT  SPWN_V_ACTIVE,SPWN_W_FLAGS(R6)  ; MARK BLOCK CURRENTLY "ACTIVE"
 66   00C0 CB   D0     04C6    680         MOVL    PRC_L_SPWN(R11),SPWN_L_LINK(R6) ; INSERT INTO LINKED LIST
00C0 CB   56   D0     04CB    681         MOVL    R6,PRC_L_SPWN(R11)              ; PASS ADDRESS OF SPWN TO TERMINATIO
                     04D0    682                                                 ; AST, SO WE CAN FIND OUT IF IT DIES
                     04D0    683
                     04D0    684 ;
                     04D0    685 ; CONSTRUCT THE PROCESS NAME OF THE FORM <USERNAME>_1.  THE TRAILING
                     04D0    686 ; NUMBER WILL BE INCREMENTED UNTIL WE FIND A UNIQUE PROCESS NAME.
                     04D0    687 ;
       01   E0     04D0    688         BBS     #SPWN_V_PRCNAM,-                ; BRANCH IF /PROCESS GIVEN
    6F 0C A6        04D2    689                 SPWN_Q_FLAGS(R6),SPAWN_PROCESS  ;
     54   01   D0     04D5    690         MOVL    #1,R4                           ; START WITH POSTFIX #1
                     04D8    691
                     04D8    692 CONSTRUCT_PRCNAM:
 53   18 A6   9E     04D8    693         MOVAB   SPWN_Q_PRCNAM(R6),R3            ; GET ADDRESS OF DESCRIPTOR SPACE
       7E   D4     04DC    694         CLRL    -(SP)                           ; MARK END OF JPI LIST
       53   DD     04DE    695         PUSHL   R3                              ; ADDRESS TO STORE RETURN LENGTH
```

```
                7E   68   7D   04E0   696          MOVQ    (R8),-(SP)                      ; DESCRIPTOR OF STRING BUFFER
        02 AE   0202 8F   B0   04E3   697          MOVW    #JPI$_USERNAME,2(SP)            ; SET PARAMETER ID TO RETRIEVE
                50   5E   D0   04E9   698          MOVL    SP,R0                           ; SET ADDRESS OF ITEM LIST
                               04EC   699          $GETJPIW_S  ITMLST=(R0),-               ; GET USERNAME STRING
                               04EC   700                      IOSB=SPWN_Q_IOSB(R6),-      ;
                               04EC   701                      EFN=#EXE$C_SYSEFN           ;
                5E   10   C0   0504   702          ADDL    #4*4,SP                         ; REMOVE ITEM LIST FROM STACK
          04 A3   04   A8   D0   0507   703          MOVL    4(R8),4(R3)                     ; SET ADDRESS OF BUFFER
        04 B3     63   20   3A   050C   704          LOCC    #^A' ',(R3),@4(R3)              ; FIND END OF NAME
                  63   50   C2   0511   705          SUBL    R0,(R3)                         ; TRUNCATE TO FIRST SPACE (IF ANY)
      7E    04    A8   63   C1   0514   706          ADDL3   (R3),4(R8),-(SP)                ; COMPUTE SPACE LEFT AFTER PRCNAM
         7E       68   63   C3   0519   707          SUBL3   (R3),(R8),-(SP)                 ;
         51    FAF3   CF   9E   051D   708          MOVAB   PRCNAM_NAME,R1                  ; GET ADDRESS OF FAO STRING
                50   81   9A   0522   709          MOVZBL  (R1)+,R0                        ; CONSTRUCT DESCRIPTOR OF STRING
                7E   50   7D   0525   710          MOVQ    R0,-(SP)                        ; PUSH DESCRIPTOR ONTO STACK
                50   5E   D0   0528   711          MOVL    SP,R0                           ; GET ADDRESS OF DESCRIPTOR
                               052B   712          $FAO_S  CTRSTR=(R0),-                   ; CONSTRUCT MAILBOX DEVICE NAME
                               052B   713                  OUTBUF=8(R0),-               ;
                               052B   714                  OUTLEN=2(R0),-               ;
                               052B   715                  P1=R3,-                      ; ADDRESS OF USERNAME DESCRIPTOR
                               052B   716                  P2=R4                        ; PROCESS NUMBER
                5E   08   C0   053E   717          ADDL    #8,SP                           ; POP FAO CONTROL STRING DESCRIPTOR
                63   8E   7D   0541   718          MOVQ    (SP)+,(R3)                      ; SET PROCESS NAME DESCRIPTOR
                               0544   719
                               0544   720 ;
                               0544   721 ; CREATE THE SUBPROCESS
                               0544   722 ;
                               0544   723 $SPAWN_PROCESS:
                51   FAB8 CF   9E   0544   724          MOVAB   LOGINOUT,R1                 ; CONSTRUCT DESCRIPTOR OF IMAGE NAME
                     50   81   9A   0549   725          MOVZBL  (R1)+,R0                    ;
             00C6 C6     50   7D   054C   726          MOVQ    R0,SPWN_Q_CLI(R6)           ; SAVE DEFAULT CLI NAME
          51   00001000 8F   D0   0551   727 5$:       MOVL    #PRC$M_CLISPEC,R1           ; ASSUME NON-INTERACTIVE (ALSO
                               0558   728                                                 ;   UNCONDITIONALLY INDICATE THAT
                               0558   729                                                 ;   CLI IS SPECIFIED.
           07 0C A6    04   E0   0558   730          BBS     #SPWN_V_MODE,SPWN_W_FLAGS(R6),6$; BRANCH IF SO
          51   00000400 8F   C8   055D   731          BISL    #PRC$M_INTER,R1             ; SET INTERACTIVE BIT
                               0564   732 6$:       $CREPRC_S IMAGE=SPWN_Q_CLI(R6),-        ; CREATE THE PROCESS
                               0564   733                  INPUT=SPWN_Q_INPUT(R6),-     ; SPECIFY INPUT
                               0564   734                  OUTPUT=SPWN_Q_OUTPUT(R6),-   ; SPECIFY OUTPUT
                               0564   735                  ERROR=SPWN_Q_MBXNAM(R6),-    ; ERROR = COMM. MAILBOX NAME
                               0564   736                  PRCNAM=SPWN_Q_PRCNAM(R6),-   ; PROCESS NAME (USERNAME_1)
                               0564   737                  BASPRI=SPWN_L_PRIB(R6),-     ; PROPAGATE BASE PRIORITY
                               0564   738                  QUOTA=SPWN_Q_QUOTAS(R6),-    ; PROPAGATE NON-DEDUCTIBLE QUOTAS
                               0564   739                  MBXUNT=SPWN_Q_UNIT(R6),-     ; TERMINATION MAILBOX
                               0564   740                  PIDADR=SPWN_L_SUBPID(R6),-   ; RECEIVE PID HERE
                               0564   741                  STSFLG=R1                    ; SET/CLEAR INTERACTIVE BIT
             0094 8F     50   B1   0590   742          CMPW    R0,#SS$_DUPLNAM             ; PROCESS NAME ALREADY EXIST?
                         0F   12   0595   743          BNEQ    10$                         ; IF NOT, PROCESS AS ANY OTHER ERROR
                         01   E0   0597   744          BBS     #SPWN_V_PRCNAM,-            ; BRANCH IF /PROCESS GIVEN
                                 0599   745                  SPWN_Q_FLAGS(R6),10$        ;
FF32 54    01    000000FF 8F   F1   059C   746          ACBL    #255,#T,R4,CONSTRUCT_PRCNAM ; INCREMENT NUMBER AFTER PROCESS NAM
                        03 50   E8   05A6   747 10$:     BLBS    R0,15$                      ; IF ERROR CREATING PROCESS, REPORT
                      0116   31   05A9   748          BRW     SPAWN_EXIT                  ; CLEANUP AND EXIT WITH STATUS
                               05AC   749
                               05AC   750 ;
                               05AC   751 ; SAVE PROCESS NAME IN SPAWN BLOCK FOR USE BY /NOTIFY.
                               05AC   752 ;
```

```
                18 A6   90  05AC   753 15$:    MOVB    SPWN_Q_PRCNAM(R6),-             ; GET LENGTH OF PROCESS NAME
              0092 C6       05AF   754                  SPWN_T_PROCESS(R6)              ;
                18 A6   28  05B2   755          MOVC3   SPWN_Q_PRCNAM(R6),-             ; COPY THE NAME
                1C B6       05B5   756                  @SPWN_Q_PRCNAM+4(R6),-          ;
              0093 C6       05B7   757                  SPWN_T_PROCESS+1(R6)            ;
                            05BA   758
                            05BA   759 ;
                            05BA   760 ; WRITE /LOG MESSAGE INDICATING PROCESS WAS CREATED
                            05BA   761 ;
                   00   E1  05BA   762          BBC     #SPWN_V_LOG,-                   ; BRANCH IF /NOLOG SPECIFIED
                10 0C A6    05BC   763                  SPWN_Q_FLAGS(R6),20$            ;
                18 A6   9F  05BF   764          PUSHAB  SPWN_Q_PRCNAM(R6)               ; ADDRESS OF PROCESS NAME
                51   01 D0  05C2   765          MOVL    #1,R1                           ; SET NUMBER OF ARGS
          50  0003FD01 8F  D0  05C5   766          MOVL    #CLI$_SPAWNED,R0                ; MESSAGE CODE
                    FA31'   30  05CC   767          BSBW    DCL$FORMMSG                     ; OUTPUT MESSAGE
                            05CF   768
                            05CF   769 ;
                            05CF   770 ; IF /WAIT /LOG THEN WRITE MESSAGE INDICATING SUBPROCESS ATTACHED.
                            05CF   771 ; DO THIS BEFORE WE WRITE THE CONTEXT, WHILE WE STILL HAVE CONTROL OVER
                            05CF   772 ; THE TERMINAL.
                            05CF   773 ;
                   02   E1  05CF   774 20$:    BBC     #SPWN_V_WAIT,-                  ; SKIP IF /NOWAIT
                15 0C A6    05D1   775                  SPWN_Q_FLAGS(R6),30$            ;
                   00   E1  05D4   776          BBC     #SPWN_V_LOG,-                   ; SKIP IF /NOLOG SPECIFIED
                10 0C A6    05D6   777                  SPWN_Q_FLAGS(R6),30$            ;
                18 A6   9F  05D9   778          PUSHAB  SPWN_Q_PRCNAM(R6)               ; ADDRESS OF PROCESS NAME
                51   01 D0  05DC   779          MOVL    #1,R1                           ; SET NUMBER OF ARGS
          50  0003FD09 8F  D0  05DF   780          MOVL    #CLI$_ATTACHED,R0               ; MESSAGE CODE
                    FA17'   30  05E6   781          BSBW    DCL$FORMMSG                     ; OUTPUT MESSAGE
                            05E9   782
                            05E9   783 ;
                            05E9   784 ; WRITE CONTEXT TO SUBPROCESS VIA MAILBOX.  IF AN ERROR OCCURS AT THIS POINT,
                            05E9   785 ; WE MUST DELETE THE PROCESS TO AVOID HAVING IT HANG AROUND FOR NOTHING.
                            05E9   786 ;
          23 0C A6    0E    E0  05E9   787 30$:    BBS     #SPWN_V_NOCTX,SPWN_W_FLAGS(R6),35$; BRANCH IF UNKNOWN CLI WAS SPECIF
                    088F    30  05EE   788          BSBW    WRITE_CONTEXT                   ; WRITE CONTEXT TO SUBPROCESS
                   1D 50    E8  05F1   789          BLBS    R0,35$                          ; BRANCH IF OK
                      50    DD  05F4   790          PUSHL   R0                              ; SAVE WRITE STATUS
                            05F6   791          $DELPRC_S PIDADR=SPWN_L_SUBPID(R6)      ; DELETE THE SUBPROCESS
             01    44 A6    D1  0602   792          CMPL    SPWN_L_STATUS(R6),#1            ; GET TERMINATION STATUS
                   04    1B  0606   793          BLEQU   32$                             ; USE IT IF SIGNIFICANT
             6E    44 A6    D0  0608   794          MOVL    SPWN_L_STATUS(R6),(SP)          ;  (NEQ TO 0 OR 1)
                   50 8ED0     060C   795 32$:    POPL    R0                              ; RESTORE STATUS CODE
                   6B    11    060F   796          BRB     60$                             ; SIGNAL THE ORIGINAL ERROR, OR THE
                            0611   797                                                    ; SUBPROCESS ABNORMAL STATUS (IF ANY
                            0611   798
                            0611   799 ;
                            0611   800 ; CLEANUP CONTEXT MAILBOX, TO ELIMINATE POOL USAGE AS SOON AS POSSIBLE.
                            0611   801 ;
                            0611   802 35$:    $DASSGN_S CHAN=SPWN_W_CHAN(R6)          ; CLEANUP MAILBOX (THUS, POOL USAGE)
                0A A6    B4 061C   803          CLRW    SPWN_W_CHAN(R6)                 ; INDICATE CHANNEL WAS "REMOVED"
                   50 01 D0  061F   804          MOVL    #1,R0                           ; IGNORE STATUS FROM DEASSIGN
                            0622   805                                                    ; WHILE WAITING FOR RE-ATTACH
                            0622   806
                            0622   807 ;
                            0622   808 ; CREATE /NOWAIT SUBPROCESS.
                            0622   809 ;
```

```
                51    01    D0  0622   810          MOVL    #1,R1                           ; SET NORMAL TERMINATION STATUS
                      02    E0  0625   811          BBS     #SPWN_V_WAIT,-                  ; BRANCH IF /NOWAIT
            03 0C A6          0627   812                  SPWN_Q_FLAGS(R6),9000$  ;
                 0095   31    062A   813          BRW     SPAWN_EXIT
                            062D   814 9000$:
                            062D   815 ;
                            062D   816 ; CREATE /WAIT SUBPROCESS.  MARK THIS (PARENT) PROCESS DETACHED FROM TERMINAL.
                            062D   817 ;
                      07    E1  062D   818          BBC     #SPWN_V_ACTIVE,-                ; BRANCH IF INACTIVE (SUBPROCESS
            4A 0C A6          062F   819                  SPWN_Q_FLAGS(R6),60$            ;     ALREADY TERMINATED)
                            0632   820          SETBIT  PRC_V_DETACHED,PRC_W_FLAGS(R11) ; MARK PROCESS DETACHED
                            0637   821
                            0637   822 ;
                            0637   823 ; WAIT FOR SUBPROCESS TO RETURN CONTROL HERE (VIA ATTACH OR TERMINATION)
                            0637   824 ;
                 0B83   30  0637   825 50$:    BSBW    CHECK_FOR_HANGUP                ; CHECK FOR HANGUP AST
                51    7E    7E  063A   826          MOVAQ   -(SP),R0                        ; ALLOCATE A TEMP. IOSB
                            063D   827          $QIOW_S FUNC=#IO$_SETMODE!IO$M_WRTATTN,-  ; SET ATTENTION AST ON MAILBOX
                            063D   828                  CHAN=PRC_Q_ATTMBX(R11),-        ;
                            063D   829                  EFN=#EXE$C_SYSEFN,-             ;
                            063D   830                  IOSB=(R0),-                     ;
                            063D   831                  P1=ATTACH_AST,-                 ; ADDRESS OF AST ROUTINE
                            063D   832                  P2=R11                          ; PASS ADDRESS OF CLI STORAGE
                5E    08    C0  0663   833          ADDL    #8,SP                           ; CLEAN UP STACK
                            0666   834          $HIBER_S                                ; HIBERNATE WAITING FOR WAKEUP
                      07    E1  066D   835          BBC     #SPWN_V_ACTIVE,-                ; BRANCH IF SUBPROCESS INACTIVE
            05 0C A6          066F   836                  SPWN_Q_FLAGS(R6),55$            ; (SUBPROCESS HAS BEEN TERMINATED)
                      0F    E0  0672   837          BBS     #PRC_V_DETACHED,-               ; BRANCH IF PARENT STILL DETACHED
            C0 68 AB          0674   838                  PRC_Q_FLAGS(R11),50$            ;
                            0677   839 55$:    CLRBIT  PRC_V_DETACHED,PRC_W_FLAGS(R11) ; MARK PROCESS ATTACHED TO TERMINAL
                            067C   840
                            067C   841 ;
                            067C   842 ; WE HAVE RETURNED FROM THE SUBPROCESS EITHER VIA A RE-ATTACH REQUEST TO
                            067C   843 ; THIS PROCESS OR VIA SUBPROCESS TERMINATION.  IF /LOG WAS SPECIFIED, THEN
                            067C   844 ; OUTPUT THE RETURNED MESSAGE.
                            067C   845 ;
                      00    E1  067C   846 60$:    BBC     #SPWN_V_LOG,-                   ; BRANCH IF /NOLOG SPECIFIED
                08 0C A6          067E   847                  SPWN_Q_FLAGS(R6),70$            ;
                      50    DD  0681   848          PUSHL   R0                              ; SAVE FINAL STATUS
                         07B0   30  0683   849          BSBW    RETURNED_MESSAGE                ; WRITE MESSAGE SAYING WE HAVE RETUR
                      50 8ED0   0686   850          POPL    R0                              ; RESTORE FINAL STATUS
                            0689   851
                            0689   852 ;
                            0689   853 ; IF WE ARE IN AN INTERACTIVE PROCESS, THEN TELL THE TERMINAL DRIVER THAT
                            0689   854 ; WE NOW OWN THE TERMINAL.
                            0689   855 ;
            28 68 AB    06    E0  0689   856 70$:    BBS     #PRC_V_MODE,PRC_W_FLAGS(R11),80$; SKIP IF NOT INTERACTIVE
                      50    DD  068E   857          PUSHL   R0                              ; SAVE FINAL STATUS
                            0690   858          $QIOW_S FUNC=#IO$_SETMODE!IO$M_TT_PROCESS,- ; ASSUME TERMINAL OWNERSHIP
                            0690   859                  CHAN=PRC_Q_INPCHAN(R11),-        ;
                            0690   860                  IOSB=SPWN_Q_IOSB(R6),-           ;
                            0690   861                  EFN=#EXE$C_SYSEFN                ;
                      50 8ED0   06B3   862          POPL    R0                              ; RESTORE FINAL STATUS
                            06B6   863
                            06B6   864 ;
                            06B6   865 ; IF THIS IS A RE-ATTACH THEN SKIP DATA STRUCTURE AND MAILBOX DELETIONS AND
                            06B6   866 ; SET NORMAL TERMINATION STATUS.  OTHERWISE, SET THE RETURNED TERMINATION STATUS.
```

```
                     06B6   867 ;
          51   01    D0 06B6   868 80$:    MOVL    #1,R1                           ; SET NORMAL TERMINATION STATUS
               07    E0 06B9   869         BBS     #SPWN_V_ACTIVE,-                ; BRANCH IF CONTROL RETURNED
          33 0C A6       06BB   870                 SPWN_B_FLAGS(R6),RESTORE_CONTEXT;    VIA A RE-ATTACH
       51    44 A6    D0 06BE   871         MOVL    SPWN_L_STATUS(R6),R1            ; GET TERMINATION STATUS
                     06C2   872
                     06C2   873 ;
                     06C2   874 ; THE SPAWN COMMAND IS ESSENTIALLY COMPLETE.  WE HAVE RETURNED CONTROL HERE
                     06C2   875 ; DUE TO SUBPROCESS TERMINATION.  THERFORE, CLEANUP MISC. DATA STRUCTURES AND
                     06C2   876 ; MAILBOXES AND REPORT ANY ERRORS WHICH WERE DETECTED.
                     06C2   877 ;
                     06C2   878 ; IF SUBPROCESS WAS SUCCESSFULLY SPAWNED AND TERMINATED, THEN TMBX WAS
                     06C2   879 ; DELETED BY THE TERMINATION AST.  IF NOT, TRY TO DEASSIGN THE MAILBOX AND
                     06C2   880 ; DELETE THE DATA STRUCTURE HERE.
                     06C2   881 ;
                     06C2   882 SPAWN_EXIT:
          7E   50    7D 06C2   883         MOVQ    R0,-(SP)                        ; SAVE ERROR AND TERMINATION STATUS
            0D 50    E8 06C5   884         BLBS    R0,10$                          ; BRANCH IF SUCCESSFUL SPAWN
               07    E1 06C8   885         BBC     #SPWN_V_ACTIVE,-                ; DID SUBPROCESS TERMINATE?
          08 0C A6       06CA   886                 SPWN_B_FLAGS(R6),10$            ; IF SO, TERMINATION MBX TAKEN CARE
             08 A7    97 06CD   887         DECB    TMBX_B_REFS(R7)                 ; DECR REFERENCE COUNT TO MAILBOX
                03    12 06D0   888         BNEQ    10$                             ; BRANCH IF STILL OUTSTANDING USES
                072A  30 06D2   889         BSBW    DELETE_TMBX                     ; DELETE TERMINATION MAILBOX AND TMB
                     06D5   890
                     06D5   891 ;
                     06D5   892 ; IN ALL CASES, DELETE THE ATTACH MAILBOX AND DEASSIGN THE CONTEXT MAILBOX
                     06D5   893 ; CHANNEL.
                     06D5   894 ;
                068A 30 06D5   895 10$:    BSBW    DELETE_ATTMBX                   ; DELETE THE ATTACH MAILBOX
                     06D8   896         $DASSGN_S CHAN=SPWN_W_CHAN(R6)              ; DEASSIGN CONTEXT MAILBOX CHANNEL
             0A A6   B4 06E3   897         CLRW    SPWN_W_CHAN(R6)                 ; INDICATE CHANNEL WAS "REMOVED"
                     06E6   898
                     06E6   899 ;
                     06E6   900 ; IF /WAIT, THEN DEALLOCATE THE SPWN BLOCK.
                     06E6   901 ;
                02   E1 06E6   902         BBC     #SPWN_V_WAIT,-                  ; BRANCH IF /NOWAIT
          03 0C A6       06E8   903                 SPWN_B_FLAGS(R6),20$            ;
                072B 30 06EB   904         BSBW    DEALLOC_SPWN                    ; REMOVE INACTIVE SPWN BLOCKS
          50    8E    7D 06EE   905 20$:    MOVQ    (SP)+,R0                        ; RESTORE ERROR AND TERMINATION STAT
                     06F1   906
                     06F1   907 ;
                     06F1   908 ; IN ALL CASES, RESTORE THE PROCESS STATE -
                     06F1   909 ;       STACK, EVENT FLAGS, CONTROL KEY AST'S, SPAWN STATUS, TERMINATION STATUS,
                     06F1   910 ;       AND SAVED REGISTERS.
                     06F1   911 ;
                     06F1   912 RESTORE_CONTEXT:
       5E  0308 CE   9E 06F1   913         MOVAB   768+8(SP),SP                    ; POP SCRATCH STORAGE OFF STACK
          7E   50    7D 06F6   914         MOVQ    R0,-(SP)                        ; SAVE ERROR AND TERMINATION STATUS
       51    58 A6    D0 06F9   915         MOVL    SPWN_L_OUTOFBAND(R6),R1         ; RESTORE OUT-OF-BAND ENABLE MASK
                F900'  30 06FD   916         BSBW    DCL$RESETOOB                    ; RESTORE AST'S
                01BF 8F BA 0700   917         POPR    #^M<R0,R1,R2,R3,R4,R5,R7,R8>    ; RESTORE REGISTERS
                     0704   918
                     0704   919 ;
                     0704   920 ; IF A HANGUP IS PENDING, THEN ABORT PROCESS
                     0704   921 ;
                0C   E1 0704   922         BBC     #PRC_V_HANGUP,-                 ; IF SET, HANG-UP PENDING
          06 68 AB       0706   923                 PRC_B_FLAGS(R11),20$            ;
```

```
5E   08   C0  0709   924          ADDL    #8,SP                   ; POP TWO PC'S OFF THE STACK
     F8F1' 31  070C   925          BRW     DCL$ABORT               ; LOG THE PROCESS OUT
           05  070F   926 20$:     RSB                             ; EXIT
```

```
                        0710   928          .SBTTL  PROCESS SPAWN INPUT STREAM
                        0710   929 ;---
                        0710   930 ;
                        0710   931 ; THIS ROUTINE IS CALLED TO PROCESS THE SPAWN INPUT STREAM.
                        0710   932 ;
                        0710   933 ; INPUTS:
                        0710   934 ;
                        0710   935 ;       R6 = ADDRESS OF SPAWN BLOCK
                        0710   936 ;       R8 = DESCRIPTOR OF SCRATCH AREA
                        0710   937 ;       R11 = ADDRESS OF PRC AREA
                        0710   938 ;
                        0710   939 ; OUTPUTS:
                        0710   940 ;
                        0710   941 ;       R0 = STATUS CODE
                        0710   942 ;---
                        0710   943
                        0710   944 VERIFY_INPUT:
                        0710   945 ;
                        0710   946 ; DETERMINE IF AN AUTOMATIC LOGOUT SHOULD OCCUR AFTER EXECUTION OF THE
                        0710   947 ; COMMAND STRING.  IF ONLY THE COMMAND STRING WAS SPECIFIED, THEN LOGOUT
                        0710   948 ; AFTER IT IS EXECUTED.  IF BOTH THE COMMAND STRING AND /INPUT WERE SPECIFIED,
                        0710   949 ; THEN CAUSE THE COMMAND STRING TO BE EXECUTED FIRST, FOLLOWED BY THE /INPUT
                        0710   950 ; STREAM.
                        0710   951 ;
              0A   E0   0710   952          BBS     #SPWN_V_INPUT,-                 ; IF INPUT SPECIFIED,
          15 0C   A6    0712   953                  SPWN_Q_FLAGS(R6),10$            ; KEEP AUTOLOGUT FLAG OFF
    20 A6   F921 CF 9A  0715   954          MOVZBL  SYS$INPUT,SPWN_Q_INPUT(R6)      ; USE "SYS$INPUT"
    24 A6   F91C CF 9E  071B   955          MOVAB   SYS$INPUT+1,SPWN_Q_INPUT+4(R6)  ;
       30   A6   B5    0721   956          TSTW    SPWN_Q_CMDSTR(R6)               ; ANY COMMAND STRING SPECIFIED?
            04   13    0724   957          BEQL    10$                             ; NO, DO NOT SET IMPLIED LOGOUT
                       0726   958          SETBIT  SPWN_V_AUTOLOGO,-               ; YES, SET IMPLIED LOGOUT FLAG
                       0726   959                  SPWN_W_FLAGS(R6)                ;
                       072A   960
                       072A   961 ;
                       072A   962 ; CHECK THAT INPUT FILE SPECIFICATION IS NOT TOO LONG.
                       072A   963 ;
50  00038200 8F   D0    072A   964 10$:    MOVL    #CLI$_INVFILSPE,R0              ; ASSUME INVALID FILE SPEC
        20   A6   B1    0731   965          CMPW    SPWN_Q_INPUT(R6),-              ; NAME TOO LONG?
        00FF 8F           0734   966                  #NAM$C_MAXRSS                   ;
             5C   1A      0737   967          BGTRU   25$                             ; IF SO, EXIT WITH ERROR
                       0739   968
                       0739   969 ;
                       0739   970 ; ALLOCATE FAB AND NAM BLOCKS AND THEN $PARSE THE INPUT FILE SPECIFICATION.
                       0739   971 ;
    5E  FF50 CE   9E    0739   972          MOVAB   -FAB$C_BLN-NAM$C_BLN(SP),SP     ; ALLOCATE FAB AND NAM BLOCKS
    00  6E   00   2C    073E   973          MOVC5   #0,(SP),#0,-                    ; ZERO FAB AND NAM
    6E  00B0 8F           0742   974                  #FAB$C_BLN+NAM$C_BLN,(SP)       ;
        54   5E   D0      0746   975          MOVL    SP,R4                           ; GET BASE OF FAB
      55   50   A4   9E    0749   976          MOVAB   FAB$C_BLN(R4),R5                ; GET BASE OF NAM
                       074D   977
                       074D   978          ASSUME  FAB$B_BLN EQ FAB$B_BID+1        ; INIT FAB
        5003 8F   B0      074D   979          MOVW    #FAB$C_BID+<FAB$C_BLN@8>,-      ; SET FAB ID
             64           0751   980                  FAB$B_BID(R4)                   ;
                       0752   981          SETBIT  FAB$V_PPF,FAB$L_FOP(R4)         ; USE PROCESS I/O SEGMENT
  34 A4   20 A6   90    0757   982          MOVB    SPWN_Q_INPUT(R6),FAB$B_FNS(R4)  ; SPECIFY FILE NAME
  2C A4   24 A6   D0    075C   983          MOVL    SPWN_Q_INPUT+4(R6),FAB$L_FNA(R4);
35 A4   F8FB CF   90    0761   984          MOVB    COM,FAB$B_DNS(R4)               ; SPECIFY DEFAULT FILE TYPE
```

```
                30 A4    F8F6 CF    9E   0767    985          MOVAB   COM+1,FAB$L_DNA(R4)                ;
                   28 A4    55     D0   076D    986          MOVL    R5,FAB$L_NAM(R4)                   ; SPECIFY NAM BLOCK ADDRESS
                                    0771    987
                                    0771    988          ASSUME  NAM$B_BLN EQ NAM$B_BID+1           ; INIT NAM
                         6002 8F    B0   0771    989          MOVW    #NAM$C_BID+<NAM$C_BLN@8>,-         ; SET NAM ID
                              65          0775    990                  NAM$B_BID(R5)                  ;
                      08 A5    10     90   0776    991          MOVB    #NAM$M_NOCONCEAL,NAM$B_NOP(R5)     ; GET PAST DOUBLE UNDERSCORES
                   0A A5    FF 8F     90   077A    992          MOVB    #NAM$C_MAXRSS,NAM$B_ESS(R5)        ; SPECIFY RESULT BUFFER
                   0C A5    04 A8     D0   077F    993          MOVL    4(R8),NAM$L_ESA(R5)                ;
                                    0784    994
                                    0784    995          $PARSE  FAB=(R4)                           ; PARSE THE FILE SPEC
                            08 50     E8   078D    996          BLBS    R0,30$                             ; CONTINUE IF SUCCESSFUL
                                    0790    997
                    5E    00B0 CE    9E   0790    998 20$:     MOVAB   FAB$C_BLN+NAM$C_BLN(SP),SP         ; DEALLOCATE FAB AND NAM
                               00EB     31   0795    999 25$:     BRW     95$                                ; EXIT WITH ERROR
                                    0798   1000
                                    0798   1001 ;
                                    0798   1002 ; PROCESS NON-PPF INPUT HERE.  IF NOT RECORD-ORIENTED, THEN USE EXPANDED
                                    0798   1003 ; STRING AS INPUT, DO A $SEARCH FOR THE FILE, AND SET SILENT LOGOUT FLAG.
                                    0798   1004 ; IF RECORD-ORIENTED THEN USE DEV FIELD AS INPUT.
                                    0798   1005 ;
                3F 34 A5    10     E0   0798   1006 30$:     BBS     #NAM$V_PPF,NAM$L_FNB(R5),50$       ; BRANCH IF PPF FILE
                   50    0B A5     9A   079D   1007          MOVZBL  NAM$B_ESL(R5),R0                   ; GET LENGTH OF RESULT STRING
                         68    50    C2   07A1   1008          SUBL    R0,(R8)                            ; PERMANENTLY REMOVE RESULT
                      04 A8    50     C0   07A4   1009          ADDL    R0,4(R8)                           ;    FROM SCRATCH AREA
                0D 40 A4    00     E1   07A8   1010          BBC     #DEV$V_REC,FAB$L_DEV(R4),40$       ; BRANCH IF NOT REC-ORIENTED
                20 A6    39 A5     9A   07AD   1011          MOVZBL  NAM$B_DEV(R5),SPWN_Q_INPUT(R6)     ; USE DEVICE AS INPUT
                24 A6    44 A5     D0   07B2   1012          MOVL    NAM$L_DEV(R5),SPWN_Q_INPUT+4(R6)   ;
                            0085     31   07B7   1013          BRW     56$                                ; PROCESS AS PPF DEVICE
                                    07BA   1014
                                    07BA   1015 ;
                                    07BA   1016 ; INPUT IS A NON-PPF NON-RECORD-ORIENTED DEVICE.  $SEARCH FOR THE FILE.
                                    07BA   1017 ; IF FOUND, THEN USE THE ESS AS SYS$INPUT.  ALSO, MARK THE PROCESS AS
                                    07BA   1018 ; NON-INTERACTIVE.
                                    07BA   1019 ;
                                    07BA   1020 40$:     $SEARCH FAB=(R4)                           ; SEARCH FOR THE SPECIFIED FILE
                         CA 50     E9   07C3   1021          BLBC    R0,20$                             ; BRANCH IF NOT FOUND
                20 A6    0B A5     9A   07C6   1022          MOVZBL  NAM$B_ESL(R5),SPWN_Q_INPUT(R6)     ; USE ESS AS INPUT FILE SPEC
                24 A6    0C A5     D0   07CB   1023          MOVL    NAM$L_ESA(R5),SPWN_Q_INPUT+4(R6)   ;
                                    07D0   1024          SETBIT  SPWN_V_MODE,SPWN_W_FLAGS(R6)       ; SET NON-INTERACTIVE SUBPROCESS
                    5E    00B0 CE    9E   07D4   1025 45$:     MOVAB   FAB$C_BLN+NAM$C_BLN(SP),SP         ; DEALLOCATE FAB AND NAM
                               0095     31   07D9   1026          BRW     90$                                ; DONE WITH PARSING
                                    07DC   1027
                                    07DC   1028 ;
                                    07DC   1029 ; INPUT IS PPF.  IF FROM A RECORD-ORIENTED DEVICE, DO $GETDVI TO GET
                                    07DC   1030 ; THE DEVICE NAME.  OTHERWISE, 1) IF EXPLICITLY SPECIFIED THEN RETURN AN
                                    07DC   1031 ; ERROR, 2) IF INTERACTIVE PROCESS THEN USE THE TERMINAL AS INPUT, 3) IF
                                    07DC   1032 ; NON-INTERACTIVE PROCESS THEN USE NL: AS INPUT.
                                    07DC   1033 ;
                   0C 0C A6    0A     E1   07DC   1034 50$:     BBC     #SPWN_V_INPUT,SPWN_W_FLAGS(R6),52$ ; SKIP IF IMPLICIT INPUT
                50    000388EA 8F     D0   07E1   1035          MOVL    #CLI$_SPWNIO,R0                    ; ASSUME ILLEGAL INPUT FILE
                   A3 40 A4    00     E1   07E8   1036          BBC     #DEV$V_REC,FAB$L_DEV(R4),20$       ; ERROR IF NOT REC-ORIENTED
                                    07ED   1037
                                    07ED   1038 ;
                                    07ED   1039 ; PROCESS LEGAL NON-RECORD-ORIENTED PPF INPUT.
                                    07ED   1040 ;
                   4D 40 A4    00     E0   07ED   1041 52$:     BBS     #DEV$V_REC,FAB$L_DEV(R4),56$       ; BRANCH IF REC-ORIENTED
```

```
                37 68 AB    06   E0  07F2  1042          BBS     #PRC_V_MODE,PRC_W_FLAGS(R11),53$; BRANCH IF NOT INTERACTIVE
              51  00000028'8F   D0  07F7  1043          MOVL    #CTL$AG_CLIDATA+PPD$T_INPDVI,R1 ; GET ADDRESS OF ASCIC DEVICE NAME
                        55   81   9A  07FE  1044          MOVZBL  (R1)+,R5                        ; GET LENGTH OF DEVICE
                        7E   54   7D  0801  1045          MOVQ    R4,-(SP)                        ; SAVE R4/R5
                04 B8   61   55   28  0804  1046          MOVC3   R5,(R1),@4(R8)                  ; COPY STRING TO SCRATCH AREA
                        63   3A   90  0809  1047          MOVB    #^A':',(R3)                     ; ADD COLON TO END OF STRING
                        54   8E   7D  080C  1048          MOVQ    (SP)+,R4                        ; RESTORE R4/R5
                             55   D6  080F  1049          INCL    R5                              ; ADD LENGTH OF A COLON
                   20 A6     55   D0  0811  1050          MOVL    R5,SPWN_Q_INPUT(R6)             ; SAVE DEVICE DESCRIPTOR
                24 A6   04   A8   D0  0815  1051          MOVL    4(R8),SPWN_Q_INPUT+4(R6)        ;
                        68   55   A2  081A  1052          SUBW    R5,(R8)                         ; PERMANENTLY REMOVE DEVICE
                   04   A8   55   C0  081D  1053          ADDL    R5,4(R8)                        ;    FROM SCRATCH AREA
                 00000044'9F   D0  0821  1054          MOVL    @#CTL$AG_CLIDATA+PPD$L_INPDEV,- ; GET INPUT DEVICE CHARS
                        40   A4       0827  1055                  FAB$L_DEV(R4)                   ;
                26 40   A4   00   E0  0829  1056          BBS     #DEV$V_REC,FAB$L_DEV(R4),58$    ; BRANCH IF REC-ORIENTED
             20 A6    F829   CF   9A  082E  1057 53$:     MOVZBL  NL,SPWN_Q_INPUT(R6)             ; SET INPUT TO NL:
             24 A6    F824   CF   9E  0834  1058          MOVAB   NL+1,SPWN_Q_INPUT+4(R6)         ;
                        40   A4   D4  083A  1059          CLRL    FAB$L_DEV(R4)                   ; CLEAR DEVICE CHARACTERISTICS
                             15   11  083D  1060          BRB     58$                             ;
                                      083F  1061
                                      083F  1062 ;
                                      083F  1063 ; PROCESS REC-ORIENTED PPF INPUT.
                                      083F  1064 ;
                        52   20 A6   7D  083F  1065 56$:     MOVQ    SPWN_Q_INPUT(R6),R2             ; COPY DESCRIPTOR OF DEVICE NAME
                             54   68   7D  0843  1066          MOVQ    (R8),R4                         ; COPY DESCRIPTOR OF SCRATCH AREA
                                097D   30  0846  1067          BSBW    GET_DEVICE                      ; GET DEVICE NAME
                        20 A6   54   7D  0849  1068          MOVQ    R4,SPWN_Q_INPUT(R6)             ; SAVE EQUIVALENCE STRING
                             68   54   A2  084D  1069          SUBW    R4,(R8)                         ; PERMANENTLY REMOVE RESULT
                        04 A8   54   C0  0850  1070          ADDL    R4,4(R8)                        ;    FROM SCRATCH AREA
                                      0854  1071
                                      0854  1072 ;
                                      0854  1073 ; GET TERMINAL CHARACTERISTICS.  DO NOT SPAWN THE SUBPROCESS IF THE TERMINAL
                                      0854  1074 ; HAS AN ASSOCIATED MAILBOX.
                                      0854  1075 ;
                     50   40 AE   D0  0854  1076 58$:     MOVL    FAB$L_DEV(SP),R0                ; GET DEVICE CHARACTERISTICS
                   5E   00B0 CE   9E  0858  1077          MOVAB   FAB$C_BLN+NAM$C_BLN(SP),SP      ; DEALLOCATE FAB AND NAM
                     06 50   02   E0  085D  1078          BBS     #DEV$V_TRM,R0,59$               ; SKIP IF INPUT IS NOT A TERMINAL
                                      0861  1079          SETBIT  SPWN_V_MODE,SPWN_W_FLAGS(R6)    ; SET NON-INTERACTIVE SUBPROCESS
                             0A   11  0865  1080          BRB     90$                             ; ALL DONE
                      51   20 A6   7E  0867  1081 59$:     MOVAQ   SPWN_Q_INPUT(R6),R1             ; GET ADDRESS OF INPUT STRING
                             0016   30  086B  1082          BSBW    CHECK_TRMMBX                    ; CHECK FOR MAILBOX
                           12 50    E9  086E  1083          BLBC    R0,95$                          ; BRANCH IF IT EXISTS
                                      0871  1084
                                      0871  1085 ;
                                      0871  1086 ; CHECK THAT INPUT FILE SPEC IS NOT TOO LONG FOR $CREPRC TO HANDLE.
                                      0871  1087 ;
                50   00038200 8F   D0  0871  1088 90$:     MOVL    #CLI$_INVFILSPE,R0              ; ASSUME SPEC IS TOO LONG
                 20 A6   00FF 8F   B1  0878  1089          CMPW    #LNM$C_NAMLENGTH,SPWN_Q_INPUT(R6)       ; IS IT REALLY?
                             03   1F  087E  1090          BLSSU   95$                             ; YES, THEN ERROR
                                      0880  1091
                                      0880  1092 ;
                                      0880  1093 ; RETURN WITH STATUS
                                      0880  1094 ;
                        50   01   D0  0880  1095          MOVL    #1,R0                           ; SET NORMAL STATUS
                                   05  0883  1096 95$:     RSB                                     ; EXIT WITH STATUS
```

```
                     0884  1098          .SBTTL  CHECK FOR ASSOCIATED TERMINAL MAILBOX
                     0884  1099 ;---
                     0884  1100 ;
                     0884  1101 ; THIS ROUTINE IS CALLED TO CHECK FOR AN ASSOCIATED TERMINAL MAILBOX.
                     0884  1102 ; SUBPROCESS IS NOT SPAWNED OR ATTACHED IF THE TERMINAL HAS AN ASSOCIATED
                     0884  1103 ; MAILBOX.
                     0884  1104 ;
                     0884  1105 ; INPUTS:
                     0884  1106 ;
                     0884  1107 ;       R1      ADDRESS OF DESCRIPTOR OF TERMINAL DEVICE NAME
                     0884  1108 ;
                     0884  1109 ; OUTPUTS:
                     0884  1110 ;
                     0884  1111 ;       R0      STATUS CODE
                     0884  1112 ;---
                     0884  1113
                     0884  1114 CHECK_TRMMBX:
7E   52   7D         0884  1115          MOVQ    R2,-(SP)                        ; SAVE REGISTERS
     7E   7C         0887  1116          CLRQ    -(SP)                           ; ALLOCATE AN IOSB
52   7E   DE         0889  1117          MOVAL   -(SP),R2                        ; ALLOCATE LONGWORD FOR CHANNEL NUMB
5E   0C   C2         088C  1118          SUBL    #12,SP                          ; ALLOCATE DEVICE CHARACTERISTICS BU
53   5E   D0         088F  1119          MOVL    SP,R3                           ; SAVE BUFFER ADDRESS
                     0892  1120          $ASSIGN_S  DEVNAM=(R1),-                ; GET A CHANNEL TO THE TERMINAL
                     0892  1121                     CHAN=(R2)                    ;
  46 50   E9         089F  1122          BLBC    R0,90$                          ; BRANCH IF ERROR
                     08A2  1123
                     08A2  1124          $QIOW_S FUNC=#IO$_SENSEMODE,-           ; GET TERMINAL CHARACTERISTICS
                     08A2  1125                  CHAN=(R2),-                     ;
                     08A2  1126                  IOSB=4(R2),-                    ;
                     08A2  1127                  EFN=#EXE$C_SYSEFN,-             ;
                     08A2  1128                  P1=(R3),-                       ;
                     08A2  1129                  P2=#12                          ;
    23 50   E9       08C2  1130          BLBC    R0,90$                          ; BRANCH IF ERROR
50  04 A2   3C       08C5  1131          MOVZWL  4(R2),R0                        ; GET IOSB STATUS
    1C 50   E9       08C9  1132          BLBC    R0,90$                          ; BRANCH IF ERROR
                     08CC  1133
                     08CC  1134          $DASSGN_S CHAN=(R2)                     ; DEASSIGN TERMINAL CHANNEL
    0F 50   E9       08D6  1135          BLBC    R0,90$                          ; BRANCH IF ERROR
                     08D9  1136
50  000388F2 8F      08D9  1137          MOVL    #CLI$_TRMMBX,R0                 ; ASSUME ASSOCIATED TERMINAL MBX
   03 08 A3   09     08E0  1138          BBS     #TT2$V_DCL_MAILBX,8(R3),90$     ; EXIT WITH ERROR IF TRUE
    50   01   D0     08E5  1139          MOVL    #1,R0                           ; SET SUCCESS
                     08E8  1140
    5E   18   C0     08E8  1141 90$:     ADDL    #8+4+12,SP                      ; RESTORE STACK
    52   8E   7D     08EB  1142          MOVQ    (SP)+,R2                        ; RESTORE REGISTERS
                 05  08EE  1143          RSB                                     ;
```

```
                        08EF  1145          .SBTTL  PROCESS SPAWN OUTPUT STREAM
                        08EF  1146 ;---
                        08EF  1147 ;
                        08EF  1148 ; THIS ROUTINE IS CALLED TO PROCESS THE SPAWN OUTPUT STREAM.
                        08EF  1149 ;
                        08EF  1150 ; INPUTS:
                        08EF  1151 ;
                        08EF  1152 ;       R6 = ADDRESS OF SPAWN BLOCK
                        08EF  1153 ;       R8 = DESCRIPTOR OF SCRATCH AREA
                        08EF  1154 ;       R11 = ADDRESS OF PRC AREA
                        08EF  1155 ;
                        08EF  1156 ; OUTPUTS:
                        08EF  1157 ;
                        08EF  1158 ;       R0 = STATUS CODE
                        08EF  1159 ;---
                        08EF  1160
                        08EF  1161 VERIFY_OUTPUT:
                        08EF  1162 ;
                        08EF  1163 ; IF OUTPUT FILE SPECIFICATION NOT SPECIFIED, THEN USE CURRENT SYS$OUTPUT
                        08EF  1164 ;
            0B   E0     08EF  1165          BBS     #SPWN_V_OUTPUT,-                ; IF OUTPUT SPECIFIED,
        0C 0C A6        08F1  1166                  SPWN_Q_FLAGS(R6),10$            ; THEN USE IT
28 A6    F74C CF    9A  08F4  1167          MOVZBL  SYS$OUTPUT,SPWN_Q_OUTPUT(R6)    ; ELSE USE "SYS$OUTPUT"
2C A6    F747 CF    9E  08FA  1168          MOVAB   SYS$OUTPUT+1,SPWN_Q_OUTPUT+4(R6);
                        0900  1169
                        0900  1170 ;
                        0900  1171 ; CHECK THAT OUTPUT FILE SPECIFICATION IS NOT TOO LONG.
                        0900  1172 ;
50   00038200 8F   D0   0900  1173 10$:     MOVL    #CLI$_INVFILSPE,R0              ; ASSUME INVALID FILE SPEC
 00FF 8F    28 A6  B1   0907  1174          CMPW    SPWN_Q_OUTPUT(R6),#NAM$C_MAXRSS ; NAME TOO LONG?
            5C     1A   090D  1175          BGTRU   25$                             ; IF SO, EXIT WITH ERROR
                        090F  1176
                        090F  1177 ;
                        090F  1178 ; ALLOCATE FAB AND NAM BLOCKS AND THEN $PARSE THE OUTPUT FILE SPECIFICATION.
                        090F  1179 ;
5E    FF50 CE   9E  090F  1180          MOVAB   -FAB$C_BLN-NAM$C_BLN(SP),SP     ; ALLOCATE FAB AND NAM BLOCKS
00    6E   00   2C  0914  1181          MOVC5   #0,(SP),#0,-                    ; ZERO FAB AND NAM
6E    00B0 8F       0918  1182                  #FAB$C_BLN+NAM$C_BLN,(SP)       ;
      54   5E   D0  091C  1183          MOVL    SP,R4                           ; GET BASE OF FAB
   55    50   A4   9E  091F  1184          MOVAB   FAB$C_BLN(R4),R5                ; GET BASE OF NAM
                        0923  1185
                        0923  1186          ASSUME  FAB$B_BLN EQ FAB$B_BID+1        ; INIT FAB
      5003 8F   B0  0923  1187          MOVW    #FAB$C_BID+<FAB$C_BLN@8>,-      ; SET FAB ID
           64       0927  1188                  FAB$B_BID(R4)                   ;
                        0928  1189          SETBIT  FAB$V_PPF,FAB$L_FOP(R4)         ; USE PROCESS I/O SEGMENT
34 A4    28 A6  90   092D  1190          MOVB    SPWN_Q_OUTPUT(R6),FAB$B_FNS(R4) ; SPECIFY FILE NAME
2C A4    2C A6  D0   0932  1191          MOVL    SPWN_Q_OUTPUT+4(R6),FAB$L_FNA(R4);
35 A4    F72A CF    90   0937  1192          MOVB    LOG,FAB$B_DNS(R4)               ; SPECIFY DEFAULT FILE TYPE
30 A4    F725 CF    9E   093D  1193          MOVAB   LOG+1,FAB$L_DNA(R4)             ;
   28 A4    55    D0   0943  1194          MOVL    R5,FAB$L_NAM(R4)                ; SPECIFY NAM BLOCK ADDRESS
                        0947  1195
                        0947  1196          ASSUME  NAM$B_BLN EQ NAM$B_BID+1        ; INIT NAM
      6002 8F   B0  0947  1197          MOVW    #NAM$C_BID+<NAM$C_BLN@8>,-      ; SET NAM ID
           65       094B  1198                  NAM$B_BID(R5)                   ;
   08 A5    10    90   094C  1199          MOVB    #NAM$M_NOCONCEAL,NAM$B_NOP(R5)  ; GET PAST DOUBLE UNDERSCORES
0A A5    FF 8F   90   0950  1200          MOVB    #NAM$C_MAXRSS,NAM$B_ESS(R5)     ; SPECIFY RESULT BUFFER
0C A5    04 A8   D0   0955  1201          MOVL    4(R8),NAM$L_ESA(R5)             ;
```

```
                        095A  1202          $PARSE  FAB=(R4)                        ; PARSE THE FILE SPEC
                  08 50 E8  0963  1203          BLBS    R0,30$                  ; CONTINUE IF SUCCESSFUL
                        0966  1204
 5E       00B0 CE    9E  0966  1205 20$:      MOVAB   FAB$C_BLN+NAM$C_BLN(SP),SP      ; DEALLOCATE FAB AND NAM
                0092  31  096B  1206 25$:      BRW     95$                             ; EXIT WITH ERROR
                        096E  1207
                        096E  1208 ;
                        096E  1209 ; PROCESS NON-PPF OUTPUT HERE.  IF NOT RECORD-ORIENTED, THEN USE EXPANDED
                        096E  1210 ; STRING AS OUTPUT.  IF RECORD-ORIENTED THEN USE DEV FIELD AS OUTPUT.
                        096E  1211 ;
28 34 A5    10  E0  096E  1212 30$:      BBS     #NAM$V_PPF,NAM$L_FNB(R5),50$    ; BRANCH IF PPF FILE
   50    0B A5  9A  0973  1213          MOVZBL  NAM$B_ESL(R5),R0                ; GET LENGTH OF RESULT STRING
      68    50  C2  0977  1214          SUBL    R0,(R8)                         ; PERMANENTLY REMOVE RESULT
   04 A8    50  C0  097A  1215          ADDL    R0,4(R8)                        ;    FROM SCRATCH AREA
0C 40 A4    00  E1  097E  1216          BBC     #DEV$V_REC,FAB$L_DEV(R4),40$    ; BRANCH IF NOT REC-ORIENTED
28 A6    39 A5  9A  0983  1217          MOVZBL  NAM$B_DEV(R5),SPWN_Q_OUTPUT(R6) ; USE DEVICE AS OUTPUT
2C A6    44 A5  D0  0988  1218          MOVL    NAM$L_DEV(R5),SPWN_Q_OUTPUT+4(R6);
          40  11  098D  1219          BRB     56$                             ; PROCESS AS PPF DEVICE
                        098F  1220
                        098F  1221 ;
                        098F  1222 ; OUTPUT IS A NON-PPF NON-RECORD-ORIENTED DEVICE.  USE THE ESS AS OUTPUT.
                        098F  1223 ;
28 A6    0B A5  9A  098F  1224 40$:      MOVZBL  NAM$B_ESL(R5),SPWN_Q_OUTPUT(R6) ; USE ESS AS OUTPUT FILE SPEC
2C A6    0C A5  D0  0994  1225          MOVL    NAM$L_ESA(R5),SPWN_Q_OUTPUT+4(R6);
          49  11  0999  1226          BRB     90$                             ; DONE WITH PARSING
                        099B  1227
                        099B  1228 ;
                        099B  1229 ; OUTPUT IS PPF.  IF FROM A RECORD-ORIENTED DEVICE, DO $GETDVI TO GET
                        099B  1230 ; THE DEVICE NAME.  OTHERWISE, IF EXPLICITLY SPECIFIED THEN RETURN AN
                        099B  1231 ; ERROR ELSE CREATE A MAILBOX THAT WILL CAUSES WRITES TO GO TO THE PARENT'S
                        099B  1232 ; OUTPUT STREAM.
                        099B  1233 ;
0E 0C A6    0B  E1  099B  1234 50$:      BBC     #SPWN_V_OUTPUT,SPWN_W_FLAGS(R6),51$ ; SKIP IF IMPLICIT OUTPUT
50    000388EA 8F  D0  09A0  1235          MOVL    #CLI$_SPWNIO,R0                 ; ASSUME ILLEGAL OUTPUT FILE
   BA 40 A4    00  E1  09A7  1236          BBC     #DEV$V_REC,FAB$L_DEV(R4),20$    ; ERROR IF NOT REC-ORIENTED
          17  11  09AC  1237          BRB     52$
                        09AE  1238
                        09AE  1239 ;
                        09AE  1240 ; IF AT CTRL/Y LEVEL, THEN DON'T USE SYS$OUTPUT.  USE THE OUTPUT STREAM
                        09AE  1241 ; SPECIFIED AT PROCESS CREATION TIME.
                        09AE  1242 ;
          0B  E1  09AE  1243 51$:      BBC     #PRC_V_YLEVEL,-                 ; IF CTRL/Y LEVEL,
      12 68 AB          09B0  1244                  PRC_Q_FLAGS(R11),52$            ; THEN SPECIAL CASE SYS$OUTPUT
       0118 CB  D0  09B3  1245          MOVL    PRC_L_OUTRABCTX(R11),-          ; GET DEVICE CHARACTERISTICS
          40 A4          09B7  1246                  FAB$L_DEV(R4)                   ;
 51    011C CB  9E  09B9  1247          MOVAB   PRC_T_OUTDVI(R11),R1            ; GET DEVICE NAME DESCRIPTOR
    50    81  9A  09BE  1248          MOVZBL  (R1)+,R0                        ;
   28 A6    50  7D  09C1  1249          MOVQ    R0,SPWN_Q_OUTPUT(R6)            ; SAVE DEVICE NAME DESCRIPTOR
                        09C5  1250
                        09C5  1251 ;
                        09C5  1252 ; PROCESS LEGAL NON-RECORD-ORIENTED OUTPUT.
                        09C5  1253 ;
05 40 A4    00  E0  09C5  1254 52$:      BBS     #DEV$V_REC,FAB$L_DEV(R4),56$    ; ERROR IF NOT REC-ORIENTED
        083B  30  09CA  1255          BSBW    CREATE_OUTMBX                   ; CREATE OUTPUT MAILBOX
          1A  11  09CD  1256          BRB     91$                             ; DO NOT SET OUTPUT BIT
                        09CF  1257
                        09CF  1258 ;
```

```
                        09CF  1259 ; PROCESS RECORD-ORIENTED OUTPUT.
                        09CF  1260 ;
    52   28 A6   7D     09CF  1261 56$:     MOVQ    SPWN_Q_OUTPUT(R6),R2            ; COPY DESCRIPTOR OF DEVICE NAME
      54   68   7D     09D3  1262          MOVQ    (R8),R4                         ; COPY DESCRIPTOR OF SCRATCH AREA
         07ED   30     09D6  1263          BSBW    GET_DEVICE                      ; GET DEVICE NAME
    28 A6    54   7D     09D9  1264          MOVQ    R4,SPWN_Q_OUTPUT(R6)            ; SAVE EQUIVALENCE STRING
      68     54   A2     09DD  1265          SUBW    R4,(R8)                         ; PERMANENTLY REMOVE RESULT
    04 A8    54   C0     09E0  1266          ADDL    R4,4(R8)                        ;    FROM SCRATCH AREA
                        09E4  1267
                        09E4  1268 ;
                        09E4  1269 ; CHECK THAT OUTPUT FILE SPEC IS NOT TOO LONG FOR $CREPRC TO HANDLE.
                        09E4  1270 ;
                        09E4  1271 90$:     SETBIT  SPWN_V_OUTPUT,SPWN_W_FLAGS(R6)  ; SET NO MAILBOX IN USE
    5E   00B0 CE   9E     09E9  1272 91$:     MOVAB   FAB$C_BLN+NAM$C_BLN(SP),SP      ; DEALLOCATE FAB AND NAM
50    00038200 8F   D0     09EE  1273          MOVL    #CLI$_INVFILSPE,R0              ; ASSUME SPEC IS TOO LONG
 28 A6   00FF 8F   B1     09F5  1274          CMPW    #LNM$C_NAMLENGTH,SPWN_Q_OUTPUT(R6)      ; IS IT REALLY?
              03   1F     09FB  1275          BLSSU   95$                             ; YES, THEN ERROR
                        09FD  1276
                        09FD  1277 ;
                        09FD  1278 ; EXIT WITH STATUS
                        09FD  1279 ;
         50   01   D0     09FD  1280          MOVL    #1,R0                           ; SET NORMAL STATUS
                   05     0A00  1281 95$:     RSB                                     ; EXIT WITH STATUS
```

```
                     0A01  1283          .SBTTL  ATTACH COMMAND
                     0A01  1284 ;+
                     0A01  1285 ; DCL$ATTACH - ATTACH COMMAND
                     0A01  1286 ;
                     0A01  1287 ; THIS ROUTINE IS CALLED TO EXECUTE THE DCL ATTACH COMMAND.  THE ATTACH
                     0A01  1288 ; COMMAND TRANSFERS EXCLUSIVE CONTROL OF THE TERMINAL FROM THIS PROCESS
                     0A01  1289 ; TO ANOTHER PROCESS BY MUTUAL COOPERATION.  A MESSAGE IS SENT TO THE
                     0A01  1290 ; "DESTINATION" PROCESS'S ATTACH REQUEST MAILBOX ASKING IF IT WANTS TO
                     0A01  1291 ; HAVE CONTROL OF THE TERMINAL.  THE DESTINATION PROCESS RESPONDS WITH
                     0A01  1292 ; A YES/NO REPLY DEPENDING ON ITS STATE.  IF IT RESPONDS WITH YES, THEN
                     0A01  1293 ; WE MARK OURSELVES DETACHED, AND WAIT FOR A RE-ATTACH REQUEST.
                     0A01  1294 ;
                     0A01  1295 ; INPUTS:
                     0A01  1296 ;
                     0A01  1297 ;       R10 = ADDRESS OF COMMAND WORK AREA
                     0A01  1298 ;       R11 = ADDRESS OF PROCESS WORK AREA
                     0A01  1299 ;
                     0A01  1300 ;
                     0A01  1301 ; OUTPUTS:
                     0A01  1302 ;
                     0A01  1303 ;       R0 = STATUS CODE
                     0A01  1304 ;-
                     0A01  1305
                     0A01  1306 DCL$ATTACH::
                     0A01  1307 ;
                     0A01  1308 ; GET THE PROCESS NAME OR PID FROM THE COMMAND LINE
                     0A01  1309 ;
          56   D4    0A01  1310          CLRL    R6                      ; MARK NO PROCESS NAME SUPPLIED YET
          58   D4    0A03  1311          CLRL    R8                      ; MARK NO PID SUPPLIED YET
     F5F8'     30    0A05  1312 40$:     BSBW    DCL$GETDVAL             ; GET NEXT TOKEN
       38 50   E9    0A08  1313          BLBC    R0,50$                  ; BRANCH IF END OF LINE
    00   55   D1    0A0B  1314          CMPL    R5,#PTR_K_COMDQUAL      ; VERB QUALIFIER?
         0A   13    0A0E  1315          BEQL    42$                     ; BRANCH IF SO
    03   55   D1    0A10  1316          CMPL    R5,#PTR_K_PARAMETR      ; PARAMETER (PROCESS NAME)?
         F0   12    0A13  1317          BNEQ    40$                     ; IF NOT, IGNORE IT
    56   51   7D    0A15  1318          MOVQ    R1,R6                   ; SAVE DESCRIPTOR OF PROCESS NAME
         EB   11    0A18  1319          BRB     40$
     F5E3'     30    0A1A  1320 42$:     BSBW    DCL$GETNVAL             ; GET QUALIFIER NUMBER
00'8F    51   91    0A1D  1321          CMPB    R1,#CLI$K_ATTA_IDEN     ; /IDENTIFICATION?
         E2   12    0A21  1322          BNEQ    40$                     ; IF NOT, IGNORE IT
    02   54   91    0A23  1323          CMPB    R4,#PTR_K_COLON         ; IS A VALUE PRESENT?
         DD   12    0A26  1324          BNEQ    40$                     ; IF NOT, USE DEFAULT
     F5D5'     30    0A28  1325          BSBW    DCL$GETDVAL             ; GET /IDENT VALUE
    52   51   7D    0A2B  1326          MOVQ    R1,R2                   ; PASS DESCRIPTOR OF VALUE STRING
    51   00   9A    0A2E  1327          MOVZBL  #PRC_K_HEX,R1           ; DEFAULT RADIX = HEX
     F5CC'     30    0A31  1328          BSBW    DCL$CNVASCBIN           ; CONVERT TO BINARY
         05   12    0A34  1329          BNEQ    45$                     ; BRANCH IF CONVERSION ERROR
    58   51   D0    0A36  1330          MOVL    R1,R8                   ; SAVE PID AWAY
         CA   11    0A39  1331          BRB     40$
                     0A3B  1332 45$:     STATUS  EXPSYN                  ; EXPRESSION SYNTAX ERROR
                05   0A42  1333          RSB
         0A   10    0A43  1334 50$:     BSBB    DCL$ATTACH2             ; PERFORM ACTUAL ATTACH OPERATION
       06 50  E9    0A45  1335          BLBC    R0,90$                  ; BRANCH IF ERROR DETECTED
     03EB     30    0A48  1336          BSBW    RETURNED_MESSAGE        ; WRITE MESSAGE SAYING WE HAVE RETURNED
    50   01   D0    0A4B  1337          MOVL    #1,R0                   ; INDICATE SUCCESS
                05   0A4E  1338 90$:     RSB                             ; RETURN WITH STATUS
```

```
                     0A4F  1340          .SBTTL  PERFORM ATTACH OPERATION
                     0A4F  1341 ;---
                     0A4F  1342 ;
                     0A4F  1343 ; ALTERNATE ENTRY POINT, USED BY ATTACH CLI CALLBACK.  NO USE OF THE
                     0A4F  1344 ; WRK AREA MUST BE MADE AFTER THIS POINT, SINCE CLI CALLBACKS OPERATE
                     0A4F  1345 ; WITHOUT ANY SUCH AREA.
                     0A4F  1346 ;
                     0A4F  1347 ; INPUTS:
                     0A4F  1348 ;
                     0A4F  1349 ;       R6/R7 = DESCRIPTOR OF PROCESS NAME
                     0A4F  1350 ;       R8 = PID OF DESTINATION PROCESS
                     0A4F  1351 ;       R11 = ADDRESS OF PRC AREA
                     0A4F  1352 ;
                     0A4F  1353 ;       EITHER THE PID OR PROCESS NAME MUST BE SPECIFIED.
                     0A4F  1354 ;
                     0A4F  1355 ; OUTPUTS:
                     0A4F  1356 ;
                     0A4F  1357 ;       R0 = FINAL STATUS
                     0A4F  1358 ;
                     0A4F  1359 ;---
                     0A4F  1360
                     0A4F  1361 DCL$ATTACH2::
                     0A4F  1362 ;
                     0A4F  1363 ; CONTROL/Y AST'S ARE DISABLED THROUGHOUT THIS COMMAND, TO ENSURE THAT
                     0A4F  1364 ; MAILBOXES WHICH ARE CREATED HERE ARE CORRECTLY DELETED, ETC.  AS A
                     0A4F  1365 ; RESULT, WE MUST PERIODICALLY CHECK THE HANGUP FLAG IN CASE A HANGUP
                     0A4F  1366 ; IS DETECTED WHILE WE ARE OPERATING.
                     0A4F  1367 ;
 00B4 CB    DD       0A4F  1368          PUSHL   PRC_L_OUTOFBAND(R11)    ; SAVE OUT-OF-BAND ENABLE MASK
      51    D4       0A53  1369          CLRL    R1                      ; DISABLE ALL OUT-OF-BAND AST'S
    F5A8'   30       0A55  1370          BSBW    DCL$RESETOOB
                     0A58  1371
                     0A58  1372 ;
                     0A58  1373 ; LOOKUP THE DESTINATION PROCESS, AND IF PRESENT, GET THE PID AND MPID
                     0A58  1374 ;
      7E    7C       0A58  1375          CLRQ    -(SP)                   ; ALLOCATE AN IOSB
      7E    7C       0A5A  1376          CLRQ    -(SP)                   ; CREATE GETJPI ITEM LIST
    F8 AE    9F       0A5C  1377          PUSHAB  -2*4(SP)                ; SET BUFFER ADDRESS
03190004 8F    DD       0A5F  1378          PUSHL   #JPI$_PID@16+4          ; REQUEST PID, SET BUFFER LENGTH
      7E    D4       0A65  1379          CLRL    -(SP)                   ; SETUP MPID REQUEST ITEM
    F8 AE    9F       0A67  1380          PUSHAB  -2*4(SP)                ; SET BUFFER ADDRESS
03250004 8F    DD       0A6A  1381          PUSHL   #JPI$_MASTER_PID@16+4   ; REQUEST MPID, SET BUFFER LENGTH
    01C0 8F    BB       0A70  1382          PUSHR   #^M<R6,R7,R8>           ; PUSH PROCESS NAME DESCRIPTOR, PID
    50    5E    D0       0A74  1383          MOVL    SP,R0                   ;
                     0A77  1384          $GETJPIW_S  ITMLST=12(R0),-     ; GET PID OF THE PROCESS
                     0A77  1385                      IOSB=40(R0),-           ;
                     0A77  1386                      EFN=#EXE$C_SYSEFN,-     ;
                     0A77  1387                      PRCNAM=(R0),-           ;
                     0A77  1388                      PIDADR=8(R0)            ;
    5E    0C    C0       0A91  1389          ADDL    #3*4,SP                 ; POP PRCNAM DESCRIPTOR, PID
      53 8ED0       0A94  1390          POPL    R3                      ; GET THE DESTINATION MPID
    5E    08    C0       0A97  1391          ADDL    #2*4,SP                 ; POINT TO RETURNED DESTINATION PID
      52 8ED0       0A9A  1392          POPL    R2                      ; GET THE DESTINATION PID
    5E    0C    C0       0A9D  1393          ADDL    #3*4,SP                 ; POP ITEM LIST
  06 50    E9       0AA0  1394          BLBC    R0,49$                  ; BRANCH IF ERROR
    50    6E    3C       0AA3  1395          MOVZWL  (SP),R0                 ; GET IOSB STATUS
  06 50    E8       0AA6  1396          BLBS    R0,55$                  ; BRANCH IF PROCESS FOUND
```

```
                5E     08   C0   0AA9  1397 49$:    ADDL    #8,SP                       ; POP THE IOSB
                     0220   31   0AAC  1398         BRW     95$                         ; REPORT ERROR
                            0AAF  1399
                            0AAF  1400 ;
                            0AAF  1401 ; MAKE SURE WE AREN'T TRYING TO ATTACH TO OURSELVES
                            0AAF  1402 ;
                      7E    7C   0AAF  1403 55$:    CLRQ    -(SP)                       ; CREATE GETJPI ITEM LIST
                      F8    AE   9F   0AB1  1404         PUSHAB  -2*4(SP)                    ; SET BUFFER ADDRESS
                03190004    8F   DD   0AB4  1405         PUSHL   #JPI$_PID@16+4              ; REQUEST OUR PID, SET BUFFER LENGTH
                      7E    D4   0ABA  1406         CLRL    -(SP)                       ; SETUP MPID REQUEST ITEM
                      F8    AE   9F   0ABC  1407         PUSHAB  -2*4(SP)                    ; SET BUFFER ADDRESS
                03250004    8F   DD   0ABF  1408         PUSHL   #JPI$_MASTER_PID@16+4       ; REQUEST OUR MPID, SET BUFFER LENGTH
                      50    5E   D0   0AC5  1409         MOVL    SP,R0                       ;
                            0AC8  1410         $GETJPIW_S  ITMLST=(R0),-           ; GET PID OF THIS PROCESS
                            0AC8  1411                     IOSB=28(R0),-               ;
                            0AC8  1412                     EFN=#EXE$C_SYSEFN           ;
                      54  8ED0   AE0  1413         POPL    R4                          ; GET OUR MPID
                      5E    08   C0   0AE3  1414         ADDL    #2*4,SP                     ; POINT TO OUR RETURNED PID
                      51  8ED0   AE6  1415         POPL    R1                          ; GET OUR PID
                      5E    0C   C0   0AE9  1416         ADDL    #3*4,SP                     ; CLEANUP STACK
                50 000388C2    8F   D0   0AEC  1417         MOVL    #CLI$_REFUSED,R0            ; ASSUME AN ERROR
                      52    51   D1   0AF3  1418         CMPL    R1,R2                       ; TRYING TO ATTACH TO OURSELF?
                      B1    13   0AF6  1419         BEQL    49$                         ; IF SO, REJECT THE OPERATION
                      54    53   D1   0AF8  1420         CMPL    R3,R4                       ; TRYING TO ATTACH TO A PROCESS OUTSIDE
                            0AFB  1421                                                 ;     OF OUR JOB?
                      AC    12   0AFB  1422         BNEQ    49$                         ; IF SO, REJECT THE OPERATION
                      5E    08   C0   0AFD  1423         ADDL    #8,SP                       ; POP THE IOSB
                            0B00  1424
                            0B00  1425 ;
                            0B00  1426 ; CREATE ATTACH$PID LOGICAL NAME FOR DESTINATION ATTACH MAILBOX
                            0B00  1427 ;
                      5E    1E   C2   0B00  1428         SUBL    #30,SP                      ; ALLOCATE ROOM FOR NAME
                            5E   DD   0B03  1429         PUSHL   SP                          ; CREATE DESCRIPTOR FOR NAME
                            1E   DD   0B05  1430         PUSHL   #30                         ;
                51    F511   CF   9E   0B07  1431         MOVAB   ATTACH_NAME,R1              ; GET ADDRESS OF ASCIC FAO STRING
                      50    81   9A   0B0C  1432         MOVZBL  (R1)+,R0                    ; CONSTRUCT DESCRIPTOR OF STRING
                      7E    50   7D   0B0F  1433         MOVQ    R0,-(SP)                    ; PUSH DESCRIPTOR ONTO STACK
                      50    5E   D0   0B12  1434         MOVL    SP,R0                       ;
                            0B15  1435         $FAO_S  CTRSTR=(R0),-               ; CONSTRUCT LOGICAL NAME
                            0B15  1436                 OUTBUF=8(R0),-              ;
                            0B15  1437                 OUTLEN=8(R0),-              ;
                            0B15  1438                 P1=R2                       ; FROM PID FOR THIS PROCESS
                      5E    08   C0   0B26  1439         ADDL    #8,SP                       ; POP FAO STRING DESCRIPTOR
                            0B29  1440
                            0B29  1441 ;
                            0B29  1442 ; GET SYS$INPUT TRANSLATION (TERMINAL DEVICE NAME).
                            0B29  1443 ; IF WE ARE AT CTRL/Y LEVEL, USE PROCESS PERMANENT SYS$INPUT INSTEAD.
                            0B29  1444 ;
                      54    10   3C   0B29  1445 60$:    MOVZWL  #ATTMBX_MAXMSG,R4           ; SET SIZE OF SCRATCH BUFFER
                      5E    54   C2   0B2C  1446         SUBL    R4,SP                       ; ALLOCATE SCRATCH BUFFER FOR STRING
                      55    5E   D0   0B2F  1447         MOVL    SP,R5                       ; CREATE DESCRIPTOR OF BUFFER
                      7E    52   7D   0B32  1448         MOVQ    R2,-(SP)                    ; SAVE R2/R3
                            0B   E1   0B35  1449         BBC     #PRC_V_YLEVEL,-             ; IF CTRL/Y LEVEL,
                         09 68   AB   0B37  1450                 PRC_B_FLAGS(R11),62$        ; THEN SPECIAL CASE SYS$OUTPUT
                53 00000028'    8F   D0   0B3A  1451         MOVL    #CTL$AG_CLIDATA+PPD$T_INPDVI,R3 ; GET ADDRESS OF ASCIC DEVICE NAME
                            05   11   0B41  1452         BRB     64$                         ; SKIP SYS$INPUT PROCESSING
                      53    F4F3   CF   9E   0B43  1453 62$:    MOVAB   SYS$INPUT,R3                ; CREATE SYS$INPUT DESCRIPTOR
```

```
                52 83   9A  0B48  1454 64$:    MOVZBL  (R3)+,R2                        ;
                0678    30  0B4B  1455         BSBW    GET_DEVICE                      ; GET INPUT DEVICE NAME
              5C 50     E9  0B4E  1456         BLBC    R0,651$                         ; BRANCH IF ERROR
                        0B51  1457 ;
                        0B51  1458 ; CHECK IF WE ARE A TERMINAL
                        0B51  1459 ;
                7E 7C   0B51  1460         CLRQ    -(SP)                           ; ALLOCATE AN IOSB
              7E 52 7D  0B53  1461         MOVQ    R2,-(SP)                        ; PUSH DEVICE DESCRIPTOR ON STACK
              51 5E DO  0B56  1462         MOVL    SP,R1                           ; GET ADDR. OF DESCRIPTOR
                7E 7C   0B59  1463         CLRQ    -(SP)                           ; CREATE AN ITEM LIST
                F8 AE 9F  0B5B  1464         PUSHAB  -2*4(SP)                        ; SET BUFFER ADDR.
          00040004 8F DD  0B5E  1465         PUSHL   #DVI$_DEVCLASS@16+4             ; REQUEST DEVICE CLASS
                50 5E DO  0B64  1466         MOVL    SP,R0                           ;
                        0B67  1467         $GETDVIW_S  DEVNAM=(R1),-               ; GET INPUT DEVICE CLASS
                        0B67  1468                     IOSB=8(R1),-
                        0B67  1469                     EFN=#EXE$C_SYSEFN,-
                        0B67  1470                     ITMLST=(R0)
              51 6E DO  0B82  1471         MOVL    (SP),R1                         ; GET DEVICE CLASS
              5E 20 C0  0B85  1472         ADDL    #4*8,SP                         ; POP GETDVI ARGS. OFF THE STACK
                22 50   E9  0B88  1473         BLBC    R0,651$                         ; REFUSE ATTACH IF ERROR
                   50   D4  0B8B  1474         CLRL    R0                              ; ASSUME IT IS A TERMINAL
          00000042 8F   51  D1  0B8D  1475         CMPL    R1,#DC$_TERM                    ; IS INPUT DEVICE A TERMINAL?
                   02   13  0B94  1476         BEQL    641$                            ; BR IF YES. FLAG OK AS IS
                   50   D6  0B96  1477         INCL    R0                              ; INDICATE NOT A TERMINAL
                52 8E 7D  0B98  1478 641$:    MOVQ    (SP)+,R2                        ; RESTORE R2/R3
                        0B9B  1479
                        0B9B  1480 ;
                        0B9B  1481 ; CHECK FOR AN ASSOCIATED TERMINAL MAILBOX.
                        0B9B  1482 ;
              7E 54 7D  0B9B  1483         MOVQ    R4,-(SP)                        ; PUSH SYS$INPUT TRANSLATION
              51 5E DO  0B9E  1484         MOVL    SP,R1                           ; GET ADDRESS OF INPUT STRING
                13 50   E8  0BA1  1485         BLBS    R0,66$                          ; DON'T CHECK FOR MAILBOX IF
                        0BA4  1486                                                 ;     NOT A TERMINAL
                FCDD    30  0BA4  1487         BSBW    CHECK_TRMMBX                    ; CHECK FOR MAILBOX
              0D 50     E8  0BA7  1488         BLBS    R0,66$                          ; BRANCH IF SUCCESS
                        0BAA  1489
                011F    31  0BAA  1490 65$:    BRW     90$                             ;
          50 000388C2 8F  DO  0BAD  1491 651$:   MOVL    #CLI$_REFUSED,R0                ; SET STATUS
                0115    31  0BB4  1492         BRW     90$                             ;
                        0BB7  1493
                        0BB7  1494 ;
                        0BB7  1495 ; CREATE ATTACH MAILBOX FOR THIS PROCESS TO RECEIVE RE-ATTACH REQUESTS.
                        0BB7  1496 ; IT IS DONE NOW, SO THAT ANY ERRORS DETECTED WILL CAUSE THE ENTIRE
                        0BB7  1497 ; OPERATION TO FAIL, RATHER THAN SENDING THE ATTACH REQUEST TO THE
                        0BB7  1498 ; OTHER PROCESS, AND THEN FINDING OUT WE CAN'T CREATE THE RE-ATTACH
                        0BB7  1499 ; MAILBOX.
                        0BB7  1500 ;
                0128    30  0BB7  1501 66$:    BSBW    CREATE_ATTMBX                   ; CREATE ATTACH MAILBOX
              ED 50     E9  0BBA  1502         BLBC    R0,65$                          ; BRANCH IF ERROR DETECTED
                        0BBD  1503
                        0BBD  1504 ;
                        0BBD  1505 ; ASSIGN CHANNEL TO ATTACH MAILBOX
                        0BBD  1506 ;
                50 7E   DE  0BBD  1507 67$:    MOVAL   -(SP),R0                        ; ADDRESS OF LOGICAL NAME DESCRIPTOR
                        0BC0  1508         $ASSIGN_S DEVNAM=4+8+ATTMBX_MAXMSG(R0),-; ASSIGN CHANNEL TO MAILBOX
                        0BC0  1509                   CHAN=(R0)                     ;
                   52 8ED0  0BCE  1510         POPL    R2                              ; GET CHANNEL OF MAILBOX
```

```
                68 50    E9  0BD1  1511          BLBC    R0,691$                         ; BRANCH IF ERROR
                             0BD4  1512
                             0BD4  1513 ;
                             0BD4  1514 ; SEND REQUEST TO DESTINATION ATTACH REQUEST MAILBOX AND GET REPLY
                             0BD4  1515 ;
              05E6    30     0BD4  1516 68$:    BSBW    CHECK_FOR_HANGUP                ; CHECK FOR HANGUP AST
           54  7E    7E     0BD7  1517         MOVAQ   -(SP),R4                        ; ALLOCATE IOSB FOR I/O REQUESTS
                             0BDA  1518         $QIOW_S FUNC=#IO$_WRITEVBLK,-           ; WRITE TO MAILBOX
                             0BDA  1519                 CHAN=R2,-                       ;
                             0BDA  1520                 EFN=#EXE$C_SYSEFN,-             ;
                             0BDA  1521                 IOSB=(R4),-                     ;
                             0BDA  1522                 P1=@12(R4),P2=8(R4)             ; RECORD = SYS$INPUT TRANSLATION
                3B 50    E9  0BFB  1523         BLBC    R0,69$                          ; BRANCH IF ERROR
           50   64    3C     0BFE  1524         MOVZWL  (R4),R0                         ; GET FINAL STATUS FROM IOSB
                35 50    E9  0C01  1525         BLBC    R0,69$                          ; BRANCH IF ERROR
           53    7E    7E     0C04  1526         MOVAQ   -(SP),R3                        ; ALLOCATE RECORD BUFFER
        04 A3    5B    D0     0C07  1527         MOVL    R11,4(R3)                       ; GET PRC AREA ADDRS.
                             0C0B  1528         $QIOW_S FUNC=#IO$_READVBLK,-            ; READ RESPONSE FROM MAILBOX
                             0C0B  1529                 CHAN=R2,-                       ;
                             0C0B  1530                 EFN=#EXE$C_SYSEFN,-             ;
                             0C0B  1531                 IOSB=(R4),-                     ;
                             0C0B  1532                 ASTADR=READ_AST,-               ;
                             0C0B  1533                 ASTPRM=R3,-                     ;
                             0C0B  1534                 P1=(R3),P2=#4                   ; TO LONGWORD BUFFER
              53 8ED0    0C30  1535         POPL    R3                              ; POP RECORD BUFFER
                03 50    E9  0C33  1536         BLBC    R0,69$                          ; BRANCH IF ERROR
           50   64    3C     0C36  1537         MOVZWL  (R4),R0                         ; GET FINAL STATUS FROM IOSB
           5E    0C    C0     0C39  1538 69$:    ADDL    #12,SP                          ; POP THE IOSB
                7C 50    E9  0C3C  1539 691$:   BLBC    R0,80$                          ; BRANCH IF ERROR
                             0C3F  1540         $DASSGN_S CHAN=R2                       ; DEASSIGN CHANNEL TO ATTACH MAILBOX
                             0C49  1541
                             0C49  1542 ;
                             0C49  1543 ; IF DESTINATION PROCESS SAYS NO TO ATTACH REQUEST, REPORT IT TO THE USER
                             0C49  1544 ;
                79 53    E9  0C49  1545         BLBC    R3,85$                          ; BRANCH IF DESTINATION PROCESS SAYS NO
                             0C4C  1546
                             0C4C  1547
                             0C4C  1548 ;
                             0C4C  1549 ; WAIT FOR PROCESS TO RETURN CONTROL HERE (FROM ATTACH OR TERMINATION).
                             0C4C  1550 ; IF WE ARE IN AN INTERACTIVE PROCESS, THEN TELL THE TERMINAL DRIVER THAT
                             0C4C  1551 ; WE NOW OWN THE TERMINAL.
                             0C4C  1552 ;
                             0C4C  1553 ;
              056E    30     0C4C  1554 70$:    BSBW    CHECK_FOR_HANGUP                ; CHECK FOR HANGUP AST
           50   7E    7E     0C4F  1555         MOVAQ   -(SP),R0                        ; CREATE A TEMP. IOSB
                             0C52  1556         $QIOW_S FUNC=#IO$_SETMODE!IO$M_WRTATTN,-  ; SET ATTN. AST ON MAILBOX
                             0C52  1557                 CHAN=PRC_Q_ATTMBX(R11),-        ;
                             0C52  1558                 EFN=#EXE$C_SYSEFN,-             ;
                             0C52  1559                 IOSB=(R0),-                     ;
                             0C52  1560                 P1=ATTACH_AST,-                 ; ADDRESS OF AST ROUTINE
                             0C52  1561                 P2=R11                          ; PASS ADDRESS OF CLI STORAGE
           5E    08    C0     0C78  1562         ADDL    #8,SP                           ; CLEAN UP STACK
                             0C7B  1563         $HIBER_S                                ; HIBERNATE WAITING FOR WAKEUP
                 0F    E0     0C82  1564         BBS     #PRC_V_DETACHED,-               ; BRANCH IF STILL DETACHED
        C5 68    AB           0C84  1565                 PRC_Q_FLAGS(R11),70$            ;
                 06    E0     0C87  1566         BBS     #PRC_V_MODE,-                   ; SKIP IF NOT INTERACTIVE
        40 68    AB           0C89  1567                 PRC_Q_FLAGS(R11),90$            ;
```

```
                     7E   7C  0CBC  1568          CLRQ    -(SP)                           ; ALLOCATE AN IOSB
                50   5E   D0  0CBE  1569          MOVL    SP,R0                           ; GET ADDRESS OF IOSB
                          0C91  1570          $QIOW_S FUNC=#IO$_SETMODE!IO$M_TT_PROCESS,- ; ASSUME TERMINAL OWNERSHIP
                          0C91  1571                  CHAN=PRC_W_INPCHAN(R11),-
                          0C91  1572                  IOSB=(R0),-
                          0C91  1573                  EFN=#EXE$C_SYSEFN
                5E   08   C0  0CB3  1574          ADDL    #8,SP                           ; RESTORE THE STACK
                50   01   D0  0CB6  1575          MOVL    #1,R0                           ; SET SUCCESSFUL
                     11   11  0CB9  1576          BRB     90$                             ; AND EXIT
                          0CBB  1577
                          0CBB  1578 ;
                          0CBB  1579 ; CLEANUP FROM ERROR DETECTED WHILE HAVING CHANNEL OPEN TO ATTACH MAILBOX
                          0CBB  1580 ;
                          0CBB  1581 80$:     $DASSGN_S CHAN=R2                       ; DEASSIGN CHANNEL TO ATTACH MAILBOX
          50   000388C2 8F   D0  0CC5  1582 85$:     MOVL    #CLI$_REFUSED,R0        ; ATTACH REQUEST REFUSED
                          0CCC  1583
                          0CCC  1584 ;
                          0CCC  1585 ; RESTORE PROCESS CONTEXT
                          0CCC  1586 ;
                5E   3E   C0  0CCC  1587 90$:     ADDL    #8+ATTMBX_MAXMSG+8+30,SP; CLEAN STACK
                56   50   D0  0CCF  1588 95$:     MOVL    R0,R6                           ; SAVE FINAL STATUS CODE
                   0144   30  0CD2  1589          BSBW    DEALLOC_SPWN                    ; REMOVE INACTIVE SPWN BLOCKS
                   008A   30  0CD5  1590          BSBW    DELETE_ATTMBX                   ; DELETE OUR ATTACH MAILBOX
                     51 8ED0  0CD8  1591          POPL    R1                              ; RESTORE OUT-OF-BAND ENABLE MASK
                   F322'  30  0CDB  1592          BSBW    DCL$RESETOOB                    ; RESTORE AST'S
                50   56   D0  0CDE  1593          MOVL    R6,R0                           ; RESTORE EXIT STATUS
                          05  0CE1  1594          RSB                                     ; EXIT
```

```
                     0CE2  1596          .SBTTL  CREATE ATTACH REQUEST MAILBOX
                     0CE2  1597 ;---
                     0CE2  1598 ;
                     0CE2  1599 ; THIS ROUTINE IS CALLED TO CREATE AN ATTACH REQUEST MAILBOX FOR
                     0CE2  1600 ; THIS PROCESS TO RECEIVE ATTACH REQUESTS FROM OTHER PROCESSES.
                     0CE2  1601 ;
                     0CE2  1602 ; INPUTS:
                     0CE2  1603 ;
                     0CE2  1604 ;       R11 = ADDRESS OF PRC AREA
                     0CE2  1605 ;
                     0CE2  1606 ; OUTPUTS:
                     0CE2  1607 ;
                     0CE2  1608 ;       R0 = STATUS CODE
                     0CE2  1609 ;---
                     0CE2  1610
                     0CE2  1611 CREATE_ATTMBX:
           52   DD   0CE2  1612          PUSHL   R2                              ; SAVE REGISTERS
           7E   7C   0CE4  1613          CLRQ    -(SP)                           ; ALLOCATE AN IOSB
                     0CE6  1614 ;
                     0CE6  1615 ; CREATE ATTACH MAILBOX AND LOGICAL NAME, SHOULD ANY OTHER PROCESS DESIRE
                     0CE6  1616 ; TO RE-ATTACH TO THIS PROCESS LATER ON.
                     0CE6  1617 ;
           7E   7C   0CE6  1618          CLRQ    -(SP)                           ; CREATE GETJPI ITEM LIST
         F8 AE   9F  0CE8  1619          PUSHAB  -2*4(SP)                        ; SET BUFFER ADDRESS
  03190004 8F   DD  0CEB  1620          PUSHL   #JPI$_PID@16+4                  ; REQUEST OUR PID, SET BUFFER LENGTH
       50   5E   D0  0CF1  1621          MOVL    SP,R0                           ;
                     0CF4  1622          $GETJPIW_S  ITMLST=(R0),-               ; GET PID OF THIS PROCESS
                     0CF4  1623                      IOSB=16(R0),-               ;
                     0CF4  1624                      EFN=#EXE$C_SYSEFN           ;
           52 8ED0   0D0C  1625          POPL    R2                              ; GET OUR PID
      5E   0C   C0   0D0F  1626          ADDL    #3*4,SP                         ; CLEANUP STACK
      5E   1E   C2   0D12  1627          SUBL    #30,SP                          ; ALLOCATE ROOM FOR NAME
           5E   DD   0D15  1628          PUSHL   SP                              ; CREATE DESCRIPTOR FOR NAME
           1E   DD   0D17  1629          PUSHL   #30
  51   F2FF CF   9E   0D19  1630          MOVAB   ATTACH_NAME,R1                  ; GET ADDRESS OF ASCIC FAO STRING
      50   81   9A   0D1E  1631          MOVZBL  (R1)+,R0                        ; CONSTRUCT DESCRIPTOR OF STRING
      7E   50   7D   0D21  1632          MOVQ    R0,-(SP)                        ; PUSH DESCRIPTOR ONTO STACK
      50   5E   D0   0D24  1633          MOVL    SP,R0
                     0D27  1634          $FAO_S  CTRSTR=(R0),-                   ; CONSTRUCT LOGICAL NAME
                     0D27  1635                  OUTBUF=8(R0),-
                     0D27  1636                  OUTLEN=8(R0),-
                     0D27  1637                  P1=R2                           ; FROM PID FOR THIS PROCESS
      5E   08   C0   0D38  1638          ADDL    #8,SP                           ; POP FAO STRING DESCRIPTOR
      50   5E   D0   0D3B  1639          MOVL    SP,R0                           ; ADDRESS OF LOGICAL NAME DESCRIPTOR
                     0D3E  1640          $CREMBX_S CHAN=PRC_W_ATTMBX(R11),-          ; CREATE ATTACH MAILBOX
                     0D3E  1641                    BUFQUO=#ATTMBX_MAXMSG,-           ; ONLY NEEDS TO HOLD 1 MESSAGE
                     0D3E  1642                    PROMSK=#^B1111T11100001111,-      ; ONLY GIVE OWNER R/W ACCESS
                     0D3E  1643                    LOGNAM=(R0)                       ;
      5E   26   C0   0D58  1644          ADDL    #30+8,SP                        ; CLEANUP STACK
      5E   08   C0   0D5B  1645 90$:     ADDL    #8,SP                           ; POP THE IOSB
           52 8ED0   0D5E  1646          POPL    R2                              ; RESTORE REGISTERS
                05   0D61  1647          RSB
```

```
                     0D62  1649          .SBTTL  DELETE ATTACH REQUEST MAILBOX
                     0D62  1650 ;---
                     0D62  1651 ;
                     0D62  1652 ; THIS ROUTINE IS CALLED TO DELETE THE ATTACH REQUEST MAILBOX
                     0D62  1653 ; FOR THIS PROCESS.  THIS IS DONE AFTER WE COME BACK FROM DETACHED
                     0D62  1654 ; STATE, SINCE AFTER THAT POINT, WE CAN NO LONGER ACCEPT ATTACH REQUESTS.
                     0D62  1655 ;
                     0D62  1656 ; INPUTS:
                     0D62  1657 ;
                     0D62  1658 ;       R11 = ADDRESS OF PRC AREA
                     0D62  1659 ;
                     0D62  1660 ; OUTPUTS:
                     0D62  1661 ;
                     0D62  1662 ;       NONE
                     0D62  1663 ;---
                     0D62  1664
                     0D62  1665 DELETE_ATTMBX:
                     0D62  1666          $DASSGN_S CHAN=PRC_W_ATTMBX(R11) ;DELETE ATTACH MAILBOX
7A AB    B4          0D6D  1667          CLRW    PRC_W_ATTMBX(R11)        ;MARK CHANNEL NO LONGER VALID
         05          0D70  1668          RSB
```

```
                     0D71  1670          .SBTTL  CREATE TERMINATION MAILBOX
                     0D71  1671 ;---
                     0D71  1672 ;
                     0D71  1673 ; THIS ROUTINE IS CALLED TO CREATE A TERMINATION MAILBOX FOR
                     0D71  1674 ; THIS PROCESS TO RECEIVE TERMINATION MESSAGE FROM UP TO FOUR OF ITS
                     0D71  1675 ; SPAWNED SUBPROCESSES.
                     0D71  1676 ;
                     0D71  1677 ; INPUTS:
                     0D71  1678 ;
                     0D71  1679 ;       R6 = ADDRESS OF SPWN DATA STRUCTURE
                     0D71  1680 ;       R8 = ADDRESS OF SCRATCH STORAGE
                     0D71  1681 ;       R11 = ADDRESS OF PRC AREA
                     0D71  1682 ;
                     0D71  1683 ; OUTPUTS:
                     0D71  1684 ;
                     0D71  1685 ;       R0 = STATUS CODE
                     0D71  1686 ;       R7 = ADDRESS OF TMBX BLOCK
                     0D71  1687 ;---
                     0D71  1688
                     0D71  1689 CREATE_TMBX:
                     0D71  1690
                     0D71  1691 ;
                     0D71  1692 ; ALLOCATE A NEW TMBX BLOCK AND LINK IT INTO THE LIST.
                     0D71  1693 ;
           51   10   3C  0D71  1694          MOVZWL  #TMBX_C_LENGTH,R1               ; LENGTH OF STORAGE TO ALLOCATE
              F289'   30  0D74  1695          BSBW    DCL$ALLDYNMEM                   ; ALLOCATE STORAGE
            7B 50     E9  0D77  1696          BLBC    R0,90$                          ; BRANCH ON ERROR
         57    52     D0  0D7A  1697          MOVL    R2,R7                           ; SET ADDRESS OF TMBX BLOCK
      0A A7    51     B0  0D7D  1698          MOVW    R1,TMBX_W_SIZE(R7)              ; STORE SIZE OF BLOCK
         08 A7    94  0D81  1699          CLRB    TMBX_B_REFS(R7)                 ; SET REFERENCE COUNT TO ZERO
      0C A7    5B     D0  0D84  1700          MOVL    R11,TMBX_L_PRC(R7)              ; STORE BASE OF PROCESS WORK AREA
            74 AB     D0  0D88  1701          MOVL    PRC_L_TMBX(R11),-               ; INSERT BLOCK INTO LINKED LIST
                  67          0D8B  1702                  TMBX_L_LINK(R7)             ;
      74 AB    57     D0  0D8C  1703          MOVL    R7,PRC_L_TMBX(R11)              ;
                     0D90  1704
                     0D90  1705 ;
                     0D90  1706 ; CREATE A TRMINATION MAILBOX THAT CAN HOLD AS MANY MESSAGES AS ALLOWED
                     0D90  1707 ; BROTHERS AND ENABLE THE TERMINATION AST.
                     0D90  1708 ;
                     0D90  1709          $CREMBX_S CHAN=TMBX_W_CHANNEL(R7),-     ; CREATE A TERMINATION MAILBOX
                     0D90  1710                    BUFQUO=#TMBX_C_MAXREFS*ACC$C_TERMLEN
           4B 50     E9  0DA8  1711          BLBC    R0,95$                          ; BRANCH ON ERROR
                     0DAB  1712          $QIOW_S FUNC=#IO$_SETMODE!IO$M_WRTATTN,- ; SET ATTENTION AST ON MAILBOX
                     0DAB  1713                  CHAN=TMBX_W_CHANNEL(R7),-       ;
                     0DAB  1714                  IOSB=SPWN_Q_IOSB(R6),-          ;
                     0DAB  1715                  EFN=#EXE$C_SYSEFN,-             ;
                     0DAB  1716                  P1=W^TERMINATION_AST,-          ; ADDRESS OF AST ROUTINE
                     0DAB  1717                  P2=R7                           ; PASS TMBX BLOCK ADDRESS
           23 50     E9  0DD0  1718          BLBC    R0,95$                          ; BRANCH IF ERROR
      50   38 A6     3C  0DD3  1719          MOVZWL  SPWN_Q_IOSB(R6),R0              ; GET IOSB STATUS
           1C 50     E9  0DD7  1720          BLBC    R0,95$                          ; BRANCH IF ERROR
                     0DDA  1721
                     0DDA  1722 ;
                     0DDA  1723 ; GET UNIT NUMBER OF TERMINATION MAILBOX
                     0DDA  1724 ;
                     0DDA  1725          $GETCHN_S CHAN=TMBX_W_CHANNEL(R7),-      ; GET MAILBOX INFORMATION
                     0DDA  1726                    PRIBUF=(R8)
```

```
                06 50    E9  0DED  1727              BLBC    R0,95$                          ; BRANCH ON ERROR
        04 A7   14 A8    B0  0DF0  1728              MOVW    DIB$W_UNIT+8(R8),TMBX_W_UNIT(R7) ; SAVE MAILBOX UNIT NUMBER
                   05    05  0DF5  1729  90$:        RSB                                     ; RETURN WITH STATUS
                         0DF6  1730
                         0DF6  1731  ;
                         0DF6  1732  ; DELETE TERMINATION MAILBOX AND TMBX DATA STRUCTURE AND THEN
                         0DF6  1733  ; RETURN WITH STATUS
                         0DF6  1734  ;
                   50    DD  0DF6  1735  95$:        PUSHL   R0                              ; SAVE STATUS
                 0004    30  0DF8  1736              BSBW    DELETE_TMBX                     ; DELETE ALLOCATED TMBX
                   50  8ED0  0DFB  1737              POPL    R0                              ; RESTORE STATUS
                   05    05  0DFE  1738              RSB                                     ; RETURN WITH STATUS
```

```
                                I 14
                     0DFF  1740          .SBTTL  DELETE TERMINATION MAILBOX
                     0DFF  1741 ;---
                     0DFF  1742 ;
                     0DFF  1743 ; THIS ROUTINE IS CALLED TO DELETE A TERMINATION MAILBOX AND ITS ASSOCIATED
                     0DFF  1744 ; TMBX DATA STRUCTURE.
                     0DFF  1745 ;
                     0DFF  1746 ; INPUTS:
                     0DFF  1747 ;
                     0DFF  1748 ;       R7 = ADDRESS OF TMBX DATA STRUCTURE
                     0DFF  1749 ;       R11 = ADDRESS OF PRC AREA
                     0DFF  1750 ;
                     0DFF  1751 ; OUTPUTS:
                     0DFF  1752 ;
                     0DFF  1753 ;       NONE
                     0DFF  1754 ;---
                     0DFF  1755
                     0DFF  1756 DELETE_TMBX:
                     0DFF  1757          $DASSGN_S CHAN=TMBX_W_CHANNEL(R7)           ; DELETE TERMINATION MAILBOX
74 AB    67    D0    0E0A  1758          MOVL    TMBX_L_LINK(R7),PRC_L_TMBX(R11)     ; REMOVE BLOCK FROM LIST
   50    57    D0    0E0E  1759          MOVL    R7,R0                               ; SET ADDRESS OF BLOCK
51    0A A7    3C    0E11  1760          MOVZWL  TMBX_W_SIZE(R7),R1                  ; GET SIZE OF BLOCK
      F1E8'    30    0E15  1761          BSBW    DCL$DEADYNMEM                       ; DEALLOCATE BLOCK
               05    0E18  1762          RSB                                         ;
                     0E19  1763
```

```
                     0E19  1765          .SBTTL  DEALLOCATE SPWN BLOCKS
                     0E19  1766 ;---
                     0E19  1767 ;
                     0E19  1768 ; THIS ROUTINE IS CALLED TO DEALLOCATE THE SPWN BLOCKS OF TERMINATED
                     0E19  1769 ; SUBPROCESSES.
                     0E19  1770 ;
                     0E19  1771 ; INPUTS:
                     0E19  1772 ;
                     0E19  1773 ;       R11 = ADDRESS OF PRC AREA
                     0E19  1774 ;
                     0E19  1775 ; OUTPUTS:
                     0E19  1776 ;
                     0E19  1777 ;       NONE
                     0E19  1778 ;---
                     0E19  1779
                     0E19  1780 DEALLOC_SPWN:
50   00C0 CB    DE   0E19  1781          MOVAL   PRC_L_SPWN(R11),R0          ; GET ADDR OF FIRST SPWN BLOCK
                     0E1E  1782          ASSUME  SPWN_C_LINK EQ 0            ;
     51   50    D0   0E1E  1783 10$:     MOVL    R0,R1                       ; SAVE ADDR OF PREVIOUS SPWN BLOCK
     50   60    D0   0E21  1784          MOVL    SPWN_L_LINK(R0),R0          ; GET NEXT SPWN BLOCK
          0F    13   0E24  1785          BEQL    90$                         ; BRANCH IF DONE
          07    E0   0E26  1786          BBS     #SPWN_V_ACTIVE,-            ; BRANCH IF STILL IN USE
        F3 0C   A0   0E28  1787                  SPWN_B_FLAGS(R0),10$        ;
          60    D0   0E2B  1788          MOVL    SPWN_L_LINK(R0),-           ; REMOVE BLOCK FROM LIST
          61         0E2D  1789                  SPWN_L_LINK(R1)             ;
51   04   A0    3C   0E2E  1790          MOVZWL  SPWN_W_SIZE(R0),R1          ; SET LENGTH OF BLOCK
        F1CB'    30   0E32  1791          BSBW    DCL$DEADYNMEM               ; DEALLOCATE BLOCK
                 05   0E35  1792 90$:     RSB
```

```
                        0E36  1794          .SBTTL  WRITE RETURNED MESSAGE
                        0E36  1795 ;---
                        0E36  1796 ;
                        0E36  1797 ; THIS ROUTINE WRITES A MESSAGE INDICATING THAT TERMINAL CONTROL
                        0E36  1798 ; HAS RETURNED TO THIS PROCESS.
                        0E36  1799 ;
                        0E36  1800 ; INPUTS:
                        0E36  1801 ;
                        0E36  1802 ;       NONE
                        0E36  1803 ;
                        0E36  1804 ; OUTPUTS:
                        0E36  1805 ;
                        0E36  1806 ;       NONE
                        0E36  1807 ;---
                        0E36  1808
                        0E36  1809 RETURNED_MESSAGE:
                        0E36  1810 ;
                        0E36  1811 ; OUTPUT MESSAGE SAYING THAT CONTROL OF THE TERMINAL HAS RETURNED
                        0E36  1812 ; TO THIS PROCESS
                        0E36  1813 ;
            5E  10  C2  0E36  1814          SUBL    #16,SP                  ;ALLOCATE SCRATCH BUFFER FOR PRCNAM
                5E  DD  0E39  1815          PUSHL   SP                      ;CREATE DESCRIPTOR OF BUFFER
                10  DD  0E3B  1816          PUSHL   #16
            50  5E  D0  0E3D  1817          MOVL    SP,R0
                7E  D4  0E40  1818          CLRL    -(SP)                   ;MARK END OF JPI LIST
                50  DD  0E42  1819          PUSHL   R0                      ;ADDRESS TO STORE RETURN LENGTH
            7E  60  7D  0E44  1820          MOVQ    (R0),-(SP)              ;DESCRIPTOR OF STRING BUFFER
    02 AE   031C  8F  B0  0E47  1821          MOVW    #JPI$_PRCNAM,2(SP)      ;SET PARAMETER ID TO RETRIEVE
                7E  7C  0E4D  1822          CLRQ    -(SP)                   ;ALLOCATE AN IOSB
            50  5E  D0  0E4F  1823          MOVL    SP,R0                   ;
                        0E52  1824          $GETJPIW_S  ITMLST=8(R0),-      ;GET CURRENT PROCESS NAME
                        0E52  1825                      IOSB=(R0),-         ;
                        0E52  1826                      EFN=#EXE$C_SYSEFN   ;
            5E  18  C0  0E6A  1827          ADDL    #6*4,SP                 ;POP ITEM LIST AND IOSB
                5E  DD  0E6D  1828          PUSHL   SP                      ;ADDRESS OF PROCESS NAME
            51  01  D0  0E6F  1829          MOVL    #1,R1                   ;SET NUMBER OF ARGS
    50  0003FD11  8F  D0  0E72  1830          MOVL    #CLI$_RETURNED,R0       ;MESSAGE CODE
                F184'  30  0E79  1831          BSBW    DCL$FORMMSG             ;OUTPUT MESSAGE
            5E  18  C0  0E7C  1832          ADDL    #8+16,SP                ;CLEAN STACK
                    05  0E7F  1833          RSB
                        0E80  1834
```

```
                        0E80  1836          .SBTTL  WRITE CONTEXT TO SUBPROCESS
                        0E80  1837 ;---
                        0E80  1838 ;
                        0E80  1839 ; THIS ROUTINE WRITES THE PROCESS CONTEXT RECORDS TO THE SUBPROCESS
                        0E80  1840 ; INITIALIZATION ROUTINE VIA THE CONTEXT MAILBOX.  EACH RECORD HAS
                        0E80  1841 ; A TYPE FIELD WHICH DISTINGUISHES THE DIFFERENT TYPES OF INFORMATION
                        0E80  1842 ; WHICH CAN BE PASSED FROM PARENT TO SUBPROCESS.
                        0E80  1843 ;
                        0E80  1844 ; INPUTS:
                        0E80  1845 ;
                        0E80  1846 ;       R6 = ADDRESS OF SPWN STORAGE
                        0E80  1847 ;
                        0E80  1848 ; OUTPUTS:
                        0E80  1849 ;
                        0E80  1850 ;       R0 = STATUS CODE
                        0E80  1851 ;---
                        0E80  1852
                        0E80  1853 WRITE_CONTEXT:
            00BC 8F     BB  0E80  1854          PUSHR   #^M<R2,R3,R4,R5,R7>     ; SAVE REGISTERS
                        0E84  1855
                        0E84  1856 ;
                        0E84  1857 ; CONSTRUCT THE HEADER CONTEXT RECORD, WHICH CONTAINS SUCH THINGS AS PROCESS
                        0E84  1858 ; PRIVILEGES, ETC.
                        0E84  1859 ;
      5E   FC00 CE      9E  0E84  1860          MOVAB   -CTX_C_MAXLEN(SP),SP            ; ALLOCATE BUFFER FOR CONTEXT RECORD
          57   5E      D0  0E89  1861          MOVL    SP,R7                           ; SAVE ADDRESS OF BUFFER
          67   00      3C  0E8C  1862          MOVZWL  #CTX_C_HEADER,CTX_W_TYPE(R7)    ; SET TYPE OF RECORD
               7E      7C  0E8F  1863          CLRQ    -(SP)                           ; SET END OF ITEM LIST, RETLEN ADDRE
            02 A7      7F  0E91  1864          PUSHAQ  CTX_Q_PROCPRIV(R7)              ; SET ADDRESS OF BUFFER
     02040008 8F      DD  0E94  1865          PUSHL   #JPI$_PROCPRIV@16!8             ; SET GETJPI ITEM CODE AND LENGTH
          50   5E      D0  0E9A  1866          MOVL    SP,R0                           ; ADDRESS OF ITEM LIST
                           0E9D  1867          $GETJPIW_S  ITMLST=(R0),-               ; GET INFORMATION
                           0E9D  1868                      IOSB=SPWN_Q_IOSB(R6),-      ;
                           0E9D  1869                      EFN=#EXE$C_SYSEFN           ;
               5E   10     C0  0EB5  1870          ADDL    #4*4,SP                     ; POP GETJPI ITEM LIST
                 0E A7     94  0EB8  1871          CLRB    CTX_B_FLAGS(R7)             ; CLEAR THE FLAGS BYTE
                           0EBB  1872
    04 0C A6     02     E1  0EBB  1873          BBC     #SPWN_V_WAIT,SPWN_W_FLAGS(R6),5$        ; BRANCH IF NOT SET
                           0EC0  1874          SETBIT  CTX_V_WAIT,CTX_B_FLAGS(R7)              ;
                           0EC4  1875
    04 0C A6     03     E1  0EC4  1876 5$:      BBC     #SPWN_V_AUTOLOGO,SPWN_W_FLAGS(R6),15$   ; BRANCH IF NOT SET
                           0EC9  1877          SETBIT  CTX_V_AUTOLOGO,CTX_B_FLAGS(R7)          ;
                           0ECD  1878
    04 0C A6     04     E1  0ECD  1879 15$:     BBC     #SPWN_V_MODE,SPWN_W_FLAGS(R6),16$       ; BRANCH IF NOT SET
                           0ED2  1880          SETBIT  CTX_V_MODE,CTX_B_FLAGS(R7)              ;
                           0ED6  1881
    04 68 AB     07     E1  0ED6  1882 16$:     BBC     #PRC_V_VERIFY,PRC_W_FLAGS(R11),161$     ; COPY VERIFICATION FLAG
                           0EDB  1883          SETBIT  CTX_V_VERIFY,CTX_B_FLAGS(R7)            ;
                           0EDF  1884
04 00AF CB     07     E1  0EDF  1885 161$:    BBC     #PRC_V_VERIMAGE,PRC_B_FLAGS2(R11),162$  ; COPY IMAGE VERIFICATION FL
                           0EE5  1886          SETBIT  CTX_V_VERIMAGE,CTX_B_FLAGS(R7)          ;
                           0EE9  1887
  12 0C A6     09     E0  0EE9  1888 162$:    BBS     #SPWN_V_PROMPT,SPWN_W_FLAGS(R6),17$     ; BRANCH IF EXPLICIT PROMTP
    50   00F0 CB     9A  0EEE  1889          MOVZBL  PRC_B_PROMPTLEN(R11),R0                 ; GET PROMPT LENGTH
      0F A7     50     90  0EF3  1890          MOVB    R0,CTX_B_PROMPTLEN(R7)                  ;
    00F1 CB     50     28  0EF7  1891          MOVC3   R0,PRC_W_PMPTCTRL(R11),-                ; GET PROMPT STRING
              10 A7         0EFC  1892                  CTX_W_PMPTCTRL(R7)                      ;
```

```
                10   11  0EFE  1893                BRB     18$                                     ;
50   00A2 C6    9A  0F00  1894 17$:                MOVZBL  SPWN_B_PROMPTLEN(R6),R0                 ; GET PROMPT LENGTH
  0F A7    50    90  0F05  1895                    MOVB    R0,CTX_B_PROMPTLEN(R7)                  ;
00A3 C6    50    28  0F09  1896                    MOVC3   R0,SPWN_Q_PMPTCTRL(R6),-                ; GET PROMPT STRING
      10 A7          0F0E  1897                            CTX_W_PMPTCTRL(R7)                      ;
                    0F10  1898                    ;
      58 A6    D0  0F10  1899 18$:                MOVL    SPWN_L_OUTOFBAND(R6),-                  ; GET OUT-OF-BAND AST MASK
      0A A7        0F13  1900                            CTX_L_OUTOFBAND(R7)                     ;
                    0F15  1901                    ;
53    33    D0  0F15  1902                    MOVL    #CTX_C_HDRLEN,R3                        ; COMPUTE SIZE OF RECORD
   0200    30  0F18  1903                    BSBW    WRITE_MAILBOX                           ; WRITE THE RECORD TO THE MAILBOX
   23 50   E9  0F1B  1904                    BLBC    R0,19$                                  ; BRANCH IF ERROR DETECTED
                    0F1E  1905                    ;
                    0F1E  1906                    ;
                    0F1E  1907                    ; WRITE THE COMMAND STRING RECORD TO THE MAILBOX
                    0F1E  1908                    ;
   30 A6    B5  0F1E  1909                    TSTW    SPWN_Q_CMDSTR(R6)                       ; ANY COMMAND STRING?
      24    13  0F21  1910                    BEQL    20$                                     ; BRANCH IF NOT
   30 A6    B1  0F23  1911                    CMPW    SPWN_Q_CMDSTR(R6),-                     ; CHECK IF BIGGER THAN OUR BUFFER
03FE 8F         0F26  1912                            #CTX_C_MAXLEN-CTX_T_CMDSTR              ;
      06    1B  0F29  1913                    BLEQU   10$                                     ; BRANCH IF OK
03FE 8F    3C  0F2B  1914                    MOVZWL  #CTX_C_MAXLEN-CTX_T_CMDSTR,-            ; ELSE, TRUNCATE STRING
   30 A6        0F2F  1915                            SPWN_Q_CMDSTR(R6)                       ;
67    01    B0  0F31  1916 10$:                MOVW    #CTX_C_CMDSTR,CTX_W_TYPE(R7)            ; SET TYPE OF RECORD
   30 A6    28  0F34  1917                    MOVC    SPWN_Q_CMDSTR(R6),-                     ;
   34 B6        0F37  1918                            @SPWN_Q_CMDSTR+4(R6),-                  ;
   02 A7        0F39  1919                            CTX_T_CMDSTR(R7)                        ;
53    57    C2  0F3B  1920                    SUBL    R7,R3                                   ; COMPUTE LENGTH OF RECORD
   01DA    30  0F3E  1921                    BSBW    WRITE_MAILBOX                           ; WRITE THE RECORD TO THE MAILBOX
   03 50   E8  0F41  1922 19$:                BLBS    R0,20$                                  ; BRANCH IF SUCCESS
   01CA    31  0F44  1923                    BRW     90$                                     ;
                    0F47  1924                    ;
                    0F47  1925                    ;
                    0F47  1926                    ; WRITE THE PROCESS LOGICAL NAMES TO THE MAILBOX
                    0F47  1927                    ;
03 0C A6   06    E0  0F47  1928 20$:                BBS     #SPWN_V_LOGNAM,SPWN_W_FLAGS(R6),30$ ; BRANCH IF REQUESTED
      0153    31  0F4C  1929                    BRW     40$                                     ;
                    0F4F  1930                    ;
                    0F4F  1931                    ;
                    0F4F  1932                    ; FIRST WRITE ALL OF THE TABLE NAMES
                    0F4F  1933                    ;
   67    05    B0  0F4F  1934 30$:                MOVW    #CTX_C_LNMTABLE,CTX_W_TYPE(R7)          ; SET LOGICAL NAME TABLE
55   00000000'GF    D0  0F52  1935                    MOVL    G^CTL$GL_LNMDIRECT,R5                   ; GET PROCESS DIRECTORY
   55    0C A5    D0  0F59  1936                    MOVL    LNMB$L_TABLE(R5),R5                     ; GET IT'S TABLE
   50    11 A5    D0  0F5D  1937                    MOVL    LNMTH$L_CHILD(R5),R0                    ; GET FIRST TABLE
      55    50    D0  0F61  1938 31$:                MOVL    R0,R5                                   ; GET NEXT TABLE
   54    09 A5    D0  0F64  1939                    MOVL    LNMTH$L_NAME(R5),R4                     ; GET POINTER TO LNMB
58 10 A4    01    E0  0F68  1940                    BBS     #LNMB$V_CONFINE,LNMB$B_FLAGS(R4),33$ ; SKIP SUBTREE IF CONFINED
   01    0B A4    91  0F6D  1941                    CMPB    LNMB$B_ACMODE(R4),#PSL$C_EXEC           ; PRIV MODE TABLE?
      4C    1B  0F71  1942                    BLEQU   32$                                     ; DON'T COPY IF SO.
04 A7    0B A4    90  0F73  1943                    MOVB    LNMB$B_ACMODE(R4),CTX_B_ACMODE(R7)      ; COPY ACMODE
      08    8B  0F78  1944                    BICB3   #LNMB$M_TABLE,-
05 A7    10 A4        0F7A  1945                            LNMB$B_FLAGS(R4),CTX_B_TFLAGS(R7)       ; COPY NAME FLAGS
08 A7    1D A5    D0  0F7E  1946                    MOVL    LNMTH$L_BYTESLM(R5),CTX_L_QUOTA(R7)     ; COPY QUOTA
   53    0C A7    DE  0F83  1947                    MOVAL   CTX_T_LNMTABLE(R7),R3                   ; PTR TO BUFFER
   51    11 A4    DE  0F87  1948                    MOVAL   LNMB$T_NAME(R4),R1                      ; PTR TO NAME
      50    61    9A  0F8B  1949                    MOVZBL  (R1),R0                                 ; LENGTH OF NAME
```

```
                50   D6   0F8E  1950          INCL    R0                                      ; INCLUDE THE COUNT
                30   BB   0F90  1951          PUSHR   #^M<R4,R5>                              ; SAVE REGS
          63    61   50   28   0F92  1952          MOVC3   R0,(R1),(R3)                            ; COPY NAME
                30   BA   0F96  1953          POPR    #^M<R4,R5>                              ; RESTORE REGS
            51    0D   A5   D0   0F98  1954          MOVL    LNMTH$L_PARENT(R5),R1                   ; PTR TO PARENT
            51    09   A1   D0   0F9C  1955          MOVL    LNMTH$L_NAME(R1),R1                     ; PTR TO PARENT'S LNMB
            51    11   A1   DE   0FA0  1956          MOVAL   LNMB$T_NAME(R1),R1                      ; PTR TO NAME
                50   61   9A   0FA4  1957          MOVZBL  (R1),R0                         ; LENGTH OF TABLE NAME
                50   D6   0FA7  1958          INCL    R0                                      ; INCLUDE THE COUNT
                30   BB   0FA9  1959          PUSHR   #^M<R4,R5>                              ; SAVE REGS
          63    61   50   28   0FAB  1960          MOVC3   R0,(R1),(R3)                            ; COPY TABLE NAME
                30   BA   0FAF  1961          POPR    #^M<R4,R5>                              ; RESTORE REGS
                53   57   C2   0FB1  1962          SUBL2   R7,R3                           ; COMPUTE SIZE OF CONTEXT RECORD
       02 A7    53   02   A3   0FB4  1963          SUBW3   #CTX_W_ENTSIZE,R3,CTX_W_ENTSIZE(R7) ; SET SIZE OF ENTRY
                   015F   30   0FB9  1964          BSBW    WRITE_MAILBOX                   ; WRITE THE RECORD TO THE MAILBOX
                  28   50   E9   0FBC  1965          BLBC    R0,92$                          ; QUIT ON ERROR
            50    11   A5   D0   0FBF  1966 32$:     MOVL    LNMTH$L_CHILD(R5),R0                    ; GET CHILD TABLE PTR
                25   12   0FC3  1967          BNEQ    310$                                    ; AND PASS IT.
            50    15   A5   D0   0FC5  1968 33$:     MOVL    LNMTH$L_SIBLING(R5),R0                  ; SIBLING TABLE?
                1F   12   0FC9  1969          BNEQ    310$                                    ; AND PASS IT.
            55    0D   A5   D0   0FCB  1970          MOVL    LNMTH$L_PARENT(R5),R5                   ; GET SIBLING OF PARENT TABLE
            50    15   A5   D0   0FCF  1971          MOVL    LNMTH$L_SIBLING(R5),R0                  ; AND PASS IT
                15   12   0FD3  1972          BNEQ    310$
                          0FD5  1973
                          0FD5  1974 ;
                          0FD5  1975 ; NOW THAT THE TABLES EXIST, WRITE ALL OF THE LOGICAL NAMES
                          0FD5  1976 ;
          67    06   D0   0FD5  1977          MOVL    #CTX_C_LNMNAME,CTX_W_TYPE(R7)   ; SET LOGICAL NAME TYPE
54  00000000'GF    D0   0FD8  1978          MOVL    G^CTL$GL_LNMHASH,R4             ; GET POINTER TO HASH TABLE
            54    64   D2   0FDF  1979          MCOML   LNMHSH$L_MASK(R4),R4                    ; GET SIZE MASK
                54   D6   0FE2  1980          INCL    R4                                      ; CONVERT TO SIZE
                00AC   31   0FE4  1981 34$:     BRW     38$                                     ; LOOP
                          0FE7  1982
                0127   31   0FE7  1983 92$:     BRW     90$                             ;HELPER BRANCH
                FF74   31   0FEA  1984 310$:    BRW     31$                             ;HELPER BRANCH
                          0FED  1985
            55    65   D0   0FED  1986 35$:     MOVL    (R5),R5                                 ; GET NEXT LINK IN CHAIN
                F2   13   0FF0  1987          BEQL    34$                                     ; GO TO NEXT CHAIN IF NONE
       F6 10 A5    01   E0   0FF2  1988          BBS     #LNMB$V_CONFINE,LNMB$B_FLAGS(R5),35$ ; SKIP NAME IF CONFINED
       F1 10 A5    03   E0   0FF7  1989          BBS     #LNMB$V_TABLE,LNMB$B_FLAGS(R5),35$ ; SKIP NAME IF TABLE
          01    0B   A5   91   0FFC  1990          CMPB    LNMB$B_ACMODE(R5),#PSL$C_EXEC   ; PRIV MODE TABLE?
                EB   1B   1000  1991          BLEQU   35$                                     ; DON'T COPY IF SO.
       04 A7    0B   A5   90   1002  1992          MOVB    LNMB$B_ACMODE(R5),CTX_B_ACMODE(R7)      ; COPY ACMODE
       05 A7    10   A5   90   1007  1993          MOVB    LNMB$B_FLAGS(R5),CTX_B_NFLAGS(R7)       ; COPY NAME FLAGS
          53    07   A7   DE   100C  1994          MOVAL   CTX_T_LNMNAME(R7),R3                    ; PTR TO BUFFER
          51    0C   A5   D0   1010  1995          MOVL    LNMB$L_TABLE(R5),R1                     ; PTR TO TABLE
          51    09   A1   D0   1014  1996          MOVL    LNMTH$L_NAME(R1),R1                     ; PTR TO TABLE'S LNMB
          50    11   A1   9A   1018  1997          MOVZBL  LNMB$T_NAME(R1),R0                      ; LENGTH OF TABLE NAME
                50   D6   101C  1998          INCL    R0                                      ; INCLUDE THE COUNT
                30   BB   101E  1999          PUSHR   #^M<R4,R5>                              ; SAVE REGS
    63    11   A1   50   28   1020  2000          MOVC3   R0,LNMB$T_NAME(R1),(R3)                 ; COPY NAME
                30   BA   1025  2001          POPR    #^M<R4,R5>                              ; RESTORE REGS
          50    11   A5   9E   1027  2002          MOVAB   LNMB$T_NAME(R5),R0                      ; PTR TO NAME
                52   D4   102B  2003          CLRL    R2                                      ; COUNT # OF XLATIONS
                0A   11   102D  2004          BRB     351$                                    ; TRY NEXT ENTRY
          0E 60    02   E0   102F  2005 350$:     BBS     #LNMX$V_XEND,LNMX$B_FLAGS(R0),352$ ; BR IF END OF LIST
                52   D6   1033  2006          INCL    R2                                      ; COUNT THIS XLATION
```

```
                50 04 A0   9E  1035  2007                MOVAB   LNMX$T_XLATION(R0),R0           ; POINT TO STRING
                51 80      9A  1039  2008 351$:           MOVZBL  (R0)+,R1                       ; GET LENGTH OF THIS STRING
                50 51      C0  103C  2009                 ADDL2   R1,R0                          ; SKIP OVER IT
                   EE      11  103F  2010                 BRB     350$                           ; TRY NEXT ENTRY
                06 A7 52   90  1041  2011 352$:           MOVB    R2,CTX_B_TRANCNT(R7)           ; SAVE COUNT
                51 11 A5   9E  1045  2012                 MOVAB   LNMB$T_NAME(R5),R1             ; ADDR OF NAME AGAIN
                   50      D6  1049  2013                 INCL    R0                             ; COUNT TERMINATION BYTE
                50 51      C2  104B  2014                 SUBL2   R1,R0                          ; COMPUTE SIZE NEEDED
             52 0400 C7    9E  104E  2015                 MOVAB   CTX_C_MAXLEN(R7),R2            ; GET END ADDR OF CTX
                52 53      C2  1053  2016                 SUBL2   R3,R2                          ; BUFFER SPACE REMAINING
                52 50      D1  1056  2017                 CMPL    R0,R2                          ; NAME ALL FIT?
                   18      1A  1059  2018                 BGTRU   36$                            ; NOPE
                   30      BB  105B  2019                 PUSHR   #^M<R4,R5>                     ; SAVE REGS
             63 61 50      28  105D  2020                 MOVC3   R0,(R1),(R3)                   ; COPY NAME AND XLATIONS
                   30      BA  1061  2021                 POPR    #^M<R4,R5>                     ; RESTORE REGS
                53 57      C2  1063  2022                 SUBL2   R7,R3                          ; COMPUTE SIZE OF CONTEXT RECORD
          02 A7 53 02      A3  1066  2023                 SUBW3   #CTX_W_ENTSIZE,R3,CTX_W_ENTSIZE(R7) ; SET SIZE OF ENTRY
                00AD       30  1068  2024                 BSBW    WRITE_MAILBOX                  ; WRITE THE RECORD TO THE MAILBOX
                75 50      E9  106E  2025                 BLBC    R0,49$                         ; QUIT ON ERROR
                   1D      11  1071  2026                 BRB     37$                            ; ONTO THE NEXT NAME
                           1073  2027 ;
                           1073  2028 ; DEAL WITH RECORDS TOO LONG FOR ONE MAILBOX MESSAGE
                           1073  2029 ; (FOR NOW, ISSUE THE "SYMBOL TOO LONG" ERROR.
                           1073  2030 ;
                           1073  2031
          18 0C A6  00     E1  1073  2032 36$:            BBC     #SPWN_V_LOG,SPWN_W_FLAGS(R6),37$ ;BRANCH IF /NOLOG SPECIFIED
                01 A1      9F  1078  2033                 PUSHAB  1(R1)                          ; PUSH DESCRIPTOR OF THE
                7E 61      9A  107B  2034                 MOVZBL  (R1),-(SP)                     ;  LOGICAL NAME
                   5E      DD  107E  2035                 PUSHL   SP                             ; ADDRESS OF THE DESCRIPTOR
                51 01      D0  1080  2036                 MOVL    #1,R1                          ; SET NUMBER OF ARGS
          50 00038218 8F   D0  1083  2037                 MOVL    #CLI$_SYMTOOLNG,R0             ; MESSAGE CODE
                   EF73'     30  108A  2038                 BSBW    DCL$FORMMSG                    ; OUTPUT MESSAGE
                5E 08        C0  108D  2039                 ADDL    #4*2,SP                        ; CLEAN STACK
                   FF5A      31  1090  2040 37$:            BRW     35$                            ; PROCESS NEXT SYMBOL
                           1093  2041
          55 00000000'GF     D0  1093  2042 38$:            MOVL    G^CTL$GL_LNMHASH,R5            ; GET ADDR OF HASHTABLE
             55 08 A544      DE  109A  2043                 MOVAL   LNMHSH$C_BUCKET-4(R5)[R4],R5   ; GET ADDR OF BUCKET
                   EE 54     F4  109F  2044                 SOBGEQ  R4,37$                         ; LOOP OVER ALL CHAINS
                           10A2  2045
                           10A2  2046 ;
                           10A2  2047 ; WRITE THE DCL GLOBAL SYMBOLS TO THE MAILBOX
                           10A2  2048 ;
          16 0C A6  05     E1  10A2  2049 40$:            BBC     #SPWN_V_CLISYM,SPWN_W_FLAGS(R6),45$ ; BRANCH IF NOT REQUESTED
             54 28 AB      9E  10A7  2050                 MOVAB   PRC_Q_GLOBAL(R11),R4           ; GET ADDRESS OF GLOBAL LISTHEAD
             04 A7 00      90  10AB  2051                 MOVB    #CTX_C_GLOBAL,CTX_B_SYMTAB(R7) ; SET WHICH SYMBOL TABLE
                0097       30  10AF  2052                 BSBW    WRITE_SYMBOLS                  ; WRITE ALL GLOBAL SYMBOLS
             54 38 AB      9E  10B2  2053                 MOVAB   PRC_Q_LOCAL(R11),R4            ; GET ADDRESS OF LOCAL LISTHEAD
             04 A7 01      90  10B6  2054                 MOVB    #CTX_C_LOCAL,CTX_B_SYMTAB(R7)  ; SET WHICH SYMBOL TABLE
                008C       30  10BA  2055                 BSBW    WRITE_SYMBOLS                  ; WRITE ALL LOCAL SYMBOLS
                           10BD  2056
                           10BD  2057 ;
                           10BD  2058 ; WRITE THE KEYPAD STATE TO THE MAILBOX
                           10BD  2059 ;
          27 0C A6  0C     E1  10BD  2060 45$:            BBC     #SPWN_V_KEYPAD,SPWN_W_FLAGS(R6),50$ ; BRANCH IF NOT REQUESTED
             54 40 AB      9E  10C2  2061                 MOVAB   PRC_Q_KEYPAD(R11),R4           ; GET ADDRESS OF KEYPAD LISTHEAD
             04 A7 02      90  10C6  2062                 MOVB    #CTX_C_KEYTABL,CTX_B_SYMTAB(R7) ; SET WHICH SYMBOL TABLE
                007C       30  10CA  2063                 BSBW    WRITE_SYMBOLS                  ; WRITE ALL KEYPAD SYMBOLS
```

```
                67    04    B0   10CD  2064          MOVW    #CTX_C_KEYSTATE,CTX_W_TYPE(R7)  ; SET TYPE OF RECORD
              52  4C AB    D0   10D0  2065          MOVL    PRC_[_LASTKEY(R11),R2           ; GET ADDR OF ASCIC KEY STATE
              51    82    9A   10D4  2066          MOVZBL  (R2)+,R1                        ; GET DESCRIPTOR
            02 A7   51    90   10D7  2067          MOVB    R1,CTX_B_KEYLENGTH(R7)          ; COPY THE LENGTH
03 A7       62    51    28   10DB  2068          MOVC    R1,(R2),CTX_T_KEYSTATE(R7)      ; COPY THE STRING
              53    57    C2   10E0  2069          SUBL    R7,R3                           ; COMPUTE LENGTH OF RECORD
                  0035    30   10E3  2070          BSBW    WRITE_MAILBOX                   ; WRITE THE RECORD TO THE MAILBOX
                28 50     E9   10E6  2071 49$:     BLBC    R0,90$                          ; BRANCH IF ERROR DETECTED
                               10E9  2072
                               10E9  2073 ;
                               10E9  2074 ; WRITE AN END-OF-FILE TO THE MAILBOX AND WAIT FOR COMPLETION, SO THAT
                               10E9  2075 ; WE DON'T DELETE THE MAILBOX BEFORE THE SUBPROCESS EVEN GETS A CHANCE
                               10E9  2076 ; TO ASSIGN A CHANNEL TO IT.
                               10E9  2077 ;
                               10E9  2078 50$:     $QIOW_S FUNC=#IO$_WRITEOF,-     ; WRITE AN EOF TO MAILBOX
                               10E9  2079                  CHAN=SPWN_W_CHAN(R6),-  ;
                               10E9  2080                  IOSB=SPWN_Q_IOSB(R6),-  ;
                               10E9  2081                  EFN=#EXE$C_SYSEFN       ;
                   04 50     E9   110A  2082          BLBC    R0,90$                  ; BRANCH IF ERROR
              50   38 A6     3C   110D  2083          MOVZWL  SPWN_Q_IOSB(R6),R0      ; GET IOSB STATUS
           5E    0400 CE     9E   1111  2084 90$:     MOVAB   CTX_C_MAXLEN(SP),SP     ; DEALLOCATE RECORD BUFFER
                 00BC 8F     BA   1116  2085          POPR    #^M<R2,R3,R4,R5,R7>     ; RESTORE REGISTERS
                               05   111A  2086          RSB
```

```
                     111B  2088          .SBTTL  WRITE RECORD TO CONTEXT MAILBOX
                     111B  2089 ;---
                     111B  2090 ;
                     111B  2091 ; WRITE A RECORD TO THE MAILBOX WITHOUT WAITING FOR A READER
                     111B  2092 ;
                     111B  2093 ; INPUTS:
                     111B  2094 ;
                     111B  2095 ;       R6 = ADDRESS OF SPWN STORAGE
                     111B  2096 ;       R7 = ADDRESS OF RECORD TO BE OUTPUT
                     111B  2097 ;       R3 = LENGTH OF RECORD TO BE OUTPUT
                     111B  2098 ;
                     111B  2099 ; OUTPUTS:
                     111B  2100 ;
                     111B  2101 ;       R0 = FINAL STATUS
                     111B  2102 ;---
                     111B  2103
                     111B  2104 WRITE_MAILBOX:
           009F   30 111B  2105          BSBW    CHECK_FOR_HANGUP                ; CHECK FOR HANGUP AST
                  111E  2106          $QIOW_S FUNC=#IO$_WRITEVBLK!IO$M_NOW,-  ; WRITE TO MAILBOX WITHOUT WAITING
                  111E  2107                  CHAN=SPWN_W_CHAN(R6),-          ;
                  111E  2108                  EFN=#EXE$C_SYSEFN,-             ;
                  111E  2109                  IOSB=SPWN_Q_IOSB(R6),-          ;
                  111E  2110                  P1=(R7),P2=R3                   ; ADDRESS/LENGTH OF BUFFER
        04 50     E9  1141  2111          BLBC    R0,90$                      ; BRANCH IF ERROR DETECTED
   50   38 A6     3C  1144  2112          MOVZWL  SPWN_Q_IOSB(R6),R0          ; GET FINAL STATUS
                 05   1148  2113 90$:     RSB
```

```
                     1149  2115          .SBTTL  WRITE ALL SYMBOLS IN A SYMBOL TABLE
                     1149  2116 ;---
                     1149  2117 ;
                     1149  2118 ;       WRITE ALL THE SYMBOL RECORDS TO THE MAILBOX FROM A
                     1149  2119 ;       SPECIFIED SYMBOL TABLE.
                     1149  2120 ;
                     1149  2121 ; INPUTS:
                     1149  2122 ;
                     1149  2123 ;       CTX_B_SYMTAB(R7) = TYPE OF SYMBOL TABLE
                     1149  2124 ;       R4 = ADDRESS OF SYMBOL TABLE LISTHEAD
                     1149  2125 ;       R6 = ADDRESS OF SPWN STORAGE
                     1149  2126 ;       R7 = ADDRESS OF CONTEXT RECORD BUFFER
                     1149  2127 ;
                     1149  2128 ; OUTPUTS:
                     1149  2129 ;
                     1149  2130 ;       R0-R5 DESTROYED.
                     1149  2131 ;---
                     1149  2132
                     1149  2133 WRITE_SYMBOLS:
          67   03    B0  1149  2134          MOVW    #CTX_C_CLISYM,CTX_W_TYPE(R7) ; SET TYPE OF RECORD
          55   54    D0  114C  2135          MOVL    R4,R5                        ; SAVE FOR END-OF-LIST CHECK
          54   64    D0  114F  2136 45$:     MOVL    (R4),R4                      ; GET NEXT SYMBOL ENTRY IN LIST
          55   54    D1  1152  2137          CMPL    R4,R5                        ; END OF LIST?
               43    13  1155  2138          BEQL    50$                          ; BRANCH IF SO
                     1157  2139          ASSUME  SYM_K_STRING EQ CTX_C_STRING
                     1157  2140          ASSUME  SYM_K_PERM EQ CTX_C_PERM
                     1157  2141          ASSUME  SYM_K_BINARY EQ CTX_C_BINARY
                     1157  2142          ASSUME  SYM_K_KEYPAD EQ CTX_C_KEYPAD
    05 A7    0A A4    90  1157  2143          MOVB    SYM_B_TYPE(R4),CTX_B_SYMTYPE(R7) ; COPY SYMBOL TYPE CODE
    06 A7    0B A4    90  115C  2144          MOVB    SYM_B_NONUNIQUE(R4),CTX_B_NONUNIQUE(R7) ; COPY UNIQUENESS POINT
       50    0C A4    9A  1161  2145          MOVZBL  SYM_T_SYMBOL(R4),R0          ; GET LENGTH OF SYMBOL NAME
       02    0A A4    91  1165  2146          CMPB    SYM_B_TYPE(R4),#SYM_K_BINARY ; BINARY VALUE?
                05    12  1169  2147          BNEQ    47$                          ; BRANCH IF NOT
             50 05    C0  116B  2148          ADDL    #4+1,R0                      ; COMPUTE LENGTH OF ASCIC NAME + VALUE
                0D    11  116E  2149          BRB     48$
    51    0D A440    9E  1170  2150 47$:     MOVAB   SYM_T_SYMBOL+1(R4)[R0],R1 ; ADDRESS OF WORD-COUNTED VALUE STRING
          51   61    3C  1175  2151          MOVZWL  (R1),R1                      ; GET LENGTH OF VALUE STRING
    50    03 A140    9E  1178  2152          MOVAB   3(R1)[R0],R0                 ; COMPUTE LENGTH OF ASCIC NAME + VALUE
    03F9  8F    50    B1  117D  2153 48$:     CMPW    R0,#<CTX_C_MAXLEN-CTX_T_SYMBOL> ; WILL SYMBOL VALUE FIT?
                17    1A  1182  2154          BGTRU   60$                          ; IF LARGER, THEN NO
                30    BB  1184  2155          PUSHR   #^M<R4,R5>                   ; SAVE REGISTERS
07 A7    0C A4    50    28  1186  2156          MOVC    R0,SYM_T_SYMBOL(R4),CTX_T_SYMBOL(R7) ; MOVE NAME + VALUE
                30    BA  118C  2157          POPR    #^M<R4,R5>                   ; RESTORE REGISTERS
          53    57    C2  118E  2158          SUBL    R7,R3                        ; COMPUTE SIZE OF RECORD
02 A7     53    02    A3  1191  2159          SUBW3   #CTX_W_ENTSIZE,R3,CTX_W_ENTSIZE(R7) ; SET SIZE OF ENTRY
                83    10  1196  2160          BSBB    WRITE_MAILBOX                ; WRITE RECORD TO MAILBOX
                B5    11  1198  2161          BRB     45$                          ; LOOP UNTIL TABLE EXHAUSTED
                      05  119A  2162 50$:     RSB
                          119B  2163
                          119B  2164 ;
                          119B  2165 ; WE HAVE ENCOUNTERED AN OVERSIZED SYMBOL.  ISSUE A WARNING MESSAGE SAYING
                          119B  2166 ; THAT WE ARE IGNORING IT AND THEN PROCEED.
                          119B  2167 ;
    AF 0C A6    00    E1  119B  2168 60$:     BBC     #SPWN_V_LOG,SPWN_W_FLAGS(R6),45$ ;BRANCH IF /NOLOG SPECIFIED
       7E    0D A4    9E  11A0  2169          MOVAB   SYM_T_SYMBOL+1(R4),-(SP); PUSH DESCRIPTOR OF THE
       7E    0C A4    9A  11A4  2170          MOVZBL  SYM_T_SYMBOL(R4),-(SP)   ;  SYMBOL NAME
             5E    DD  11A8  2171          PUSHL   SP                           ; ADDRESS OF THE DESCRIPTOR
```

```
                51    01    D0  11AA  2172          MOVL    #1,R1                           ; SET NUMBER OF ARGS
        50  00038218 8F    D0  11AD  2173          MOVL    #CLI$_SYMTOOLNG,R0              ; MESSAGE CODE
                EE49'   30    11B4  2174          BSBW    DCL$FORMMSG                     ; OUTPUT MESSAGE
                5E    08    C0  11B7  2175          ADDL    #4*2,SP                         ; CLEAN STACK
                    FF92    31  11BA  2176          BRW     45$                             ; PROCESS NEXT SYMBOL
                            11BD  2177
```

```
                     11BD  2179          .SBTTL  CHECK FOR PENDING HANGUP AST
                     11BD  2180 ;---
                     11BD  2181 ;
                     11BD  2182 ; THIS ROUTINE IS CALLED TO CHECK IF A HANGUP AST HAS COME IN SINCE
                     11BD  2183 ; WE STARTED THE COMMAND EXECUTION.  IF SO, THE PROCESS IS TERMINATED.
                     11BD  2184 ;
                     11BD  2185 ; INPUTS:
                     11BD  2186 ;
                     11BD  2187 ;       R11 = ADDRESS OF PRC AREA
                     11BD  2188 ;
                     11BD  2189 ; OUTPUTS:
                     11BD  2190 ;
                     11BD  2191 ;       NONE
                     11BD  2192 ;---
                     11BD  2193
                     11BD  2194 CHECK_FOR_HANGUP:
03 68 AB    0C   E1  11BD  2195          BBC     #PRC_V_HANGUP,PRC_W_FLAGS(R11),90$ ; BRANCH IF NO HANGUP PENDING
          EE3B'  31  11C2  2196          BRW     DCL$RESTART                        ; ABORT PROCESS
                 05  11C5  2197 90$:     RSB
```

```
                     11C6  2199          .SBTTL  GET DEVICE NAME
                     11C6  2200 ;---
                     11C6  2201 ;
                     11C6  2202 ; THIS ROUTINE CALLS THE $GETDVI SYSTEM SERVICE TO GET THE CANNONICAL NAME OF
                     11C6  2203 ; A SPECIFIED DEVICE.  IT PLACES THE DEVICE NAME IN THE SPECIFIED BUFFER AND
                     11C6  2204 ; PROPAGATES THE STATUS RETURNED BY $GETDVI TO THE CALLER.
                     11C6  2205 ;
                     11C6  2206 ; INPUTS:
                     11C6  2207 ;
                     11C6  2208 ;       R2/R3 = DESCRIPTOR OF DEVICE NAME
                     11C6  2209 ;       R4/R5 = DESCRIPTOR OF RETURN BUFFER
                     11C6  2210 ;
                     11C6  2211 ; OUTPUTS:
                     11C6  2212 ;
                     11C6  2213 ;       R0 = STATUS RETURNED BY $GETDVI
                     11C6  2214 ;
                     11C6  2215 ;---
                     11C6  2216
                     11C6  2217 GET_DEVICE:
        7E   7C      11C6  2218          CLRQ    -(SP)                       ; ALLOCATE AN IOSB
    7E  52   7D      11C8  2219          MOVQ    R2,-(SP)                    ; PUSH DEVICE DESCRIPTOR ON STACK
    51  5E   D0      11CB  2220          MOVL    SP,R1                       ; GET ADDRESS OF DESCIPTOR
        7E   D4      11CE  2221          CLRL    -(SP)                       ; CREATE GETDVI ITEM LIST
      F4 AE  9F      11D0  2222          PUSHAB  -12(SP)                     ; RETURN LENGTH ON STACK
    7E  54   7D      11D3  2223          MOVQ    R4,-(SP)                    ; PUSH BUFFER DESCRIPTOR ON STACK
 02 AE  20   B0      11D6  2224          MOVW    #DVI$_DEVNAM,2(SP)          ; REQUEST COMPLETE DEVICE NAME
    50  5E   D0      11DA  2225          MOVL    SP,R0                       ; GET ADDRESS OF ITEM LIST
                     11DD  2226          $GETDVIW_S  DEVNAM=(R1),-           ; GET TERMINAL DEVICE NAME
                     11DD  2227                      IOSB=8(R1),-            ;
                     11DD  2228                      EFN=#EXE$C_SYSEFN,-     ;
                     11DD  2229                      ITMLST=(R0)             ;
54  6E   3C         11F8  2230          MOVZWL  (SP),R4                     ; GET LENGTH OF RESULT
5E  18   C0         11FB  2231          ADDL    #6*4,SP                     ; RESTORE STACK
  03 50   E9         11FE  2232          BLBC    R0,90$                      ; BRANCH IF ERROR
50  6E   3C         1201  2233          MOVZWL  (SP),R0                     ; GET IOSB STATUS
5E  08   C0         1204  2234 90$:     ADDL    #8,SP                       ; POP THE IOSB
             05      1207  2235          RSB
```

```
                     1208  2237          .SBTTL  CREATE OUTPUT MAILBOX
                     1208  2238 ;---
                     1208  2239 ;
                     1208  2240 ; THIS ROUTINE IS CALLED TO CREATE AN OUTPUT MAILBOX FOR THIS PROCESS TO
                     1208  2241 ; RECEIVE WRITE REQUESTS FROM ITS SPAWNED SUBPROCESSES.
                     1208  2242 ;
                     1208  2243 ; INPUTS:
                     1208  2244 ;
                     1208  2245 ;       R6 = ADDRESS OF SPWN DATA STRUCTURE
                     1208  2246 ;       R8 = ADDRESS OF SCRATCH AREA
                     1208  2247 ;       R11 = ADDRESS OF PRC DATA STRUCTURE
                     1208  2248 ;
                     1208  2249 ; OUTPUTS:
                     1208  2250 ;
                     1208  2251 ;       R0 = STATUS CODE
                     1208  2252 ;---
                     1208  2253
                     1208  2254 CREATE_OUTMBX:
                     1208  2255
                     1208  2256 ;
                     1208  2257 ; GET PID OF PARENT.
                     1208  2258 ;
          7E   7C    1208  2259          CLRQ    -(SP)                           ; CREATE GETJPI ITEM LIST
        F8 AE   9F   120A  2260          PUSHAB  -8(SP)                          ; SET BUFFER ADDRESS
 03190004 8F   DD    120D  2261          PUSHL   #JPI$_PID@16+4                  ; REQUEST OUR PID, SET BUFFER LENGTH
       50 5E   D0    1213  2262          MOVL    SP,R0                           ; GET ADDR OF ITEM LIST
                     1216  2263          $GETJPIW_S  ITMLST=(R0),-               ; GET PID OF THIS PROCESS
                     1216  2264                      EFN=#EXE$C_SYSEFN,-         ;
                     1216  2265                      IOSB=SPWN_Q_IOSB(R6)        ;
       50 8ED0       122E  2266          POPL    R0                              ; GET OUR PID
    5E 0C   C0       1231  2267          ADDL    #3*4,SP                         ; CLEANUP STACK
                     1234  2268
                     1234  2269 ;
                     1234  2270 ; CREATE MAILBOX LOGICAL NAME.
                     1234  2271 ;
       7E   68   7D  1234  2272          MOVQ    (R8),-(SP)                      ; PUSH LOG NAME DESCRIPTOR
 52    EDF0 CF   9E  1237  2273          MOVAB   OUTPUT_NAME,R2                  ; GET ADDRESS OF ASCIC FAO STRING
       51   82   9A  123C  2274          MOVZBL  (R2)+,R1                        ; CONSTRUCT DESCRIPTOR OF STRING
       7E   51   7D  123F  2275          MOVQ    R1,-(SP)                        ; PUSH DESCRIPTOR ONTO STACK
       51   5E   D0  1242  2276          MOVL    SP,R1                           ; GET ADDR OF ARGS
                     1245  2277          $FAO_S  CTRSTR=(R1),-                   ; CONSTRUCT LOGICAL NAME
                     1245  2278                  OUTBUF=8(R1),-                  ;
                     1245  2279                  OUTLEN=8(R1),-                  ;
                     1245  2280                  P1=R0                           ; FROM PID FOR THIS PROCESS
       5E   08   C0  1256  2281          ADDL    #8,SP                           ; POP THE FAO DESCRIPTOR
    28 A6   8E   7D  1259  2282          MOVQ    (SP)+,SPWN_Q_OUTPUT(R6)         ; GET LOGICAL NAME DESC
    68   28 A6   C2  125D  2283          SUBL    SPWN_Q_OUTPUT(R6),(R8)          ; PERMANENTLY REMOVE FROM SCRATCH
04 A8   28 A6   C0  1261  2284          ADDL    SPWN_Q_OUTPUT(R6),4(R8)         ; PERMANENTLY REMOVE FROM SCRATCH
      06 50   E8  1266  2285          BLBS    R0,20$                          ; BRANCH IF SUCCESSFUL
        009B  31  1269  2286          BRW     91$                             ; BRANCH IF ERROR DETECTED
        0091  31  126C  2287 10$:     BRW     90$                             ; EXIT SUCCESSFULLY
                     126F  2288
                     126F  2289 ;
                     126F  2290 ; CHECK FOR ALREADY EXISTING MAILBOX.
                     126F  2291 ;
    00CE CB   B6  126F  2292 20$:     INCW    PRC_W_OUTMBXREF(R11)            ; INCR THE MAILBOX REF COUNT
01  00CE CB   B1  1273  2293          CMPW    PRC_W_OUTMBXREF(R11),#1         ; DOES MAILBOX ALREADY EXIST?
```

```
                F2   12  1278  2294          BNEQ    10$                                     ; YES, THEN SKIP MBX CREATION
                                127A  2295
                                127A  2296 ;
                                127A  2297 ; GET MAXIMUM MESSAGE SIZE.
                                127A  2298 ;
                7E   7C  127A  2299 25$:      CLRQ    -(SP)                                   ; CREATE THE ITEM LIST
              F8 AE   9F  127C  2300          PUSHAB  -8(SP)                                  ; SET THE BUFFER ADDRESS
      104F0004 8F   DD  127F  2301          PUSHL   #SYI$_MAXBUF@16+4                       ; SET THE ITEM CODE AND BUFFER SIZE
            50  5E   D0  1285  2302          MOVL    SP,R0                                   ; GET ADDRESS OF ITEM LIST
                            1288  2303          $GETSYIW_S  ITMLST=(R0),-                       ; GET MAX BUF SIZE
                            1288  2304                      EFN=#EXE$C_SYSEFN,-             ;
                            1288  2305                      IOSB=SPWN_Q_IOSB(R6)            ;
                51 8ED0  12A0  2306          POPL    R1                                      ; GET THE MAXMSG SIZE
             5E  0C   C0  12A3  2307          ADDL    #3*4,SP                                 ; POP ITEM LIST
                5F 50   E9  12A6  2308          BLBC    R0,95$                                  ; RETURN IF ERROR
            50  38 A6   3C  12A9  2309          MOVZWL  SPWN_Q_IOSB(R6),R0                      ; GET IOSB STATUS
                58 50   E9  12AD  2310          BLBC    R0,95$                                  ; RETURN IF ERROR
      00CC CB   51   B0  12B0  2311          MOVW    R1,PRC_W_OUTMBXSIZ(R11)                 ; SAVE THE MAXMSG SIZE
                            12B5  2312
                            12B5  2313 ;
                            12B5  2314 ; CREATE THE OUTPUT MAILBOX.
                            12B5  2315 ;
                            12B5  2316          $CREMBX_S CHAN=PRC_W_OUTMBXCHN(R11),-           ; CREATE A WRITE MAILBOX
                            12B5  2317                    MAXMSG=R1,-                       ;
                            12B5  2318                    BUFQUO=R1,-                       ;
                            12B5  2319                    LOGNAM=SPWN_Q_OUTPUT(R6)          ;
             38 50   E9  12CD  2320          BLBC    R0,95$                                  ; BRANCH ON ERROR
                            12D0  2321
                            12D0  2322 ;
                            12D0  2323 ; SET WRITE ATTENTION AST ON THE MAILBOX.
                            12D0  2324 ;
                            12D0  2325          $QIOW_S FUNC=#IO$_SETMODE!IO$M_WRTATTN,- ; SET ATTENTION AST ON MAILBOX
                            12D0  2326                  CHAN=PRC_W_OUTMBXCHN(R11),-     ;
                            12D0  2327                  EFN=#EXE$C_SYSEFN,-             ;
                            12D0  2328                  IOSB=SPWN_Q_IOSB(R6),-          ;
                            12D0  2329                  P1=W^WRITE_AST,-                ; ADDRESS OF AST ROUTINE
                            12D0  2330                  P2=R11                          ; PASS PRC BLOCK ADDRESS
             0F 50   E9  12F6  2331          BLBC    R0,95$                                  ; BRANCH IF ERROR
      50     38 A6   3C  12F9  2332          MOVZWL  SPWN_Q_IOSB(R6),R0                      ; GET IOSB STATUS
             08 50   E9  12FD  2333          BLBC    R0,95$                                  ; BRANCH IF ERROR
                            1300  2334 90$:       STATUS  NORMAL                                  ; SET NORMAL STATUS
                05  1307  2335 91$:       RSB                                             ; RETURN
                            1308  2336
                            1308  2337 ;
                            1308  2338 ; DELETE WRITE MAILBOX AND RETURN WITH STATUS
                            1308  2339 ;
                50   DD  1308  2340 95$:       PUSHL   R0                                      ; SAVE STATUS
              0005   30  130A  2341          BSBW    DELETE_OUTMBX                           ; DELETE ALLOCATED MAILBOX
                50 8ED0  130D  2342          POPL    R0                                      ; RESTORE STATUS
                F5   11  1310  2343          BRB     91$                                     ; EXIT
```

```
                     1312  2345          .SBTTL  DELETE WRITE MAILBOX
                     1312  2346 ;---
                     1312  2347 ;
                     1312  2348 ; THIS ROUTINE IS CALLED TO DELETE A WRITE MAILBOX.
                     1312  2349 ;
                     1312  2350 ; INPUTS:
                     1312  2351 ;
                     1312  2352 ;       R11 = ADDRESS OF PRC AREA
                     1312  2353 ;
                     1312  2354 ; OUTPUTS:
                     1312  2355 ;
                     1312  2356 ;       NONE
                     1312  2357 ;---
                     1312  2358
                     1312  2359 DELETE_OUTMBX:
00CE CB   B7         1312  2360          DECW    PRC_W_OUTMBXREF(R11)            ; DECR THE MAILBOX REF COUNT
      10   12        1316  2361          BNEQ    90$                             ; SKIP IF NON-ZERO
                     1318  2362          $DASSGN_S CHAN=PRC_W_OUTMBXCHN(R11)     ; DELETE OUTPUT MAILBOX
00CA CB   B4         1324  2363          CLRW    PRC_W_OUTMBXCHN(R11)            ; CLEAR MAILBOX CHANNEL NUMBER
                     1328  2364 90$:     RSB                                     ;
```

```
                     1329  2366          .SBTTL  WRITE REQUEST AST FROM A SUBPROCESS
                     1329  2367 ;---
                     1329  2368 ;
                     1329  2369 ; THIS AST ROUTINE HANDLES A WRITE REQUEST FROM A SUBPROCESS
                     1329  2370 ; THROUGH THE WRITE MAILBOX ASSOCIATED WITH THIS PROCESS.
                     1329  2371 ;
                     1329  2372 ; INPUTS:
                     1329  2373 ;
                     1329  2374 ;       4(AP) = ADDRESS OF PRC AREA
                     1329  2375 ;
                     1329  2376 ; OUTPUTS:
                     1329  2377 ;
                     1329  2378 ;       NONE
                     1329  2379 ;---
                     1329  2380
                     1329  2381 WRITE_AST:
                 087C 1329  2382          .WORD   ^M<R2,R3,R4,R5,R6,R11>
                     132B  2383
          5B  04 AC  D0  132B  2384          MOVL    4(AP),R11                       ; GET ADDRESS OF CLI PROCESS WORK AR
                     132F  2385
                     132F  2386 ;
                     132F  2387 ; IF HANGUP AST IS PENDING, THEN WAKE UP SO THAT WE CAN TERMINATE.
                     132F  2388 ;
       7A 68 AB  0C  E0  132F  2389          BBS     #PRC_V_HANGUP,PRC_W_FLAGS(R11),80$ ; WAKE UP IF HANGUP PENDING
                     1334  2390
                     1334  2391 ;
                     1334  2392 ; RE-ENABLE WRITE ATTENTION AST.
                     1334  2393 ;
             7E  7C  1334  2394          CLRQ    -(SP)                           ; ALLOCATE AN IOSB
          54 5E  D0  1336  2395          MOVL    SP,R4                           ;
                     1339  2396          $QIOW_S FUNC=#IO$_SETMODE!IO$M_WRTATTN,- ; RE-ENABLE AST
                     1339  2397                  CHAN=PRC_W_OUTMBXCHN(R11),-      ;
                     1339  2398                  EFN=#EXE$C_SYSEFN,-              ;
                     1339  2399                  IOSB=(R4),-                      ;
                     1339  2400                  P1=WRITE_AST,-                   ; ADDRESS OF AST ROUTINE
                     1339  2401                  P2=R11                          ; ADDRESS OF PRC DATA STRUCTURE
                     135D  2402
                     135D  2403 ;
                     135D  2404 ; ALLOCATE A RECORD BUFFER AND IOSB.
                     135D  2405 ;
       55 00CC CB  3C  135D  2406          MOVZWL  PRC_W_OUTMBXSIZ(R11),R5         ; GET SIZE OF RECORD BUFFER
          5E 55 C2  1362  2407          SUBL    R5,SP                           ; ALLOCATE RECORD BUFFER
          52 5E D0  1365  2408          MOVL    SP,R2                           ; GET THE BUFFER ADDRESS
                     1368  2409
                     1368  2410 ;
                     1368  2411 ; GET RECORD TO BE WRITTEN FROM THE MAILBOX.  IF THERE IS MORE THAN
                     1368  2412 ; ONE MESSAGE IN THE MAILBOX, LOOP UNTIL WE'VE READ AND WRITTEN THEM ALL.
                     1368  2413 ;
                     1368  2414 10$:     $QIOW_S FUNC=#IO$_READVBLK!IO$M_NOW,-    ; READ THE ATTACH MAILBOX
                     1368  2415                  CHAN=PRC_W_OUTMBXCHN(R11),-      ;
                     1368  2416                  EFN=#EXE$C_SYSEFN,-              ;
                     1368  2417                  IOSB=(R4),-                      ;
                     1368  2418                  P1=(R2),-                        ; ADDRESS OF BUFFER
                     1368  2419                  P2=R5                            ; SIZE OF BUFFER
          1F 50 E9  138B  2420          BLBC    R0,30$                          ; BRANCH IF DSW ERROR
          09 64 E9  138E  2421          BLBC    (R4),20$                        ; BRANCH IF I/O ERROR
       51 02 A4  32  1391  2422          CVTWL   2(R4),R1                        ; GET LENGTH OF MESSAGE
```

```
                                                M 15
                EC68'  30  1395  2423          BSBW    DCL$SPAWNOUT                    ; WRITE THE MESSAGE
                  CE   11  1398  2424          BRB     10$                             ; LOOP UNTIL MAILBOX CLEANED OUT
                       139A  2425
0870 8F           64   B1  139A  2426 20$:     CMPW    (R4),#SS$_ENDOFFILE             ; WAS ERROR EOF?
                  C7   12  139F  2427          BNEQ    10$                             ; NO, DISREGARD AND READ AGAIN
              04  A4   D5  13A1  2428          TSTL    4(R4)                           ; WAS PID SPECIFIED?
                  C2   12  13A4  2429          BNEQ    10$                             ; YES, THEN KEEP READING TO EMPTY
                       13A6  2430          STATUS  NORMAL                          ; SET NORMAL STATUS
                       04  13AD  2431 30$:     RET                             ; RETURN
                       13AE  2432
                       13AE  2433 80$:     $WAKE_S                         ; WAKE UP CURRENT PROCESS
                       04  13B9  2434          RET
```

```
                        13BA  2436          .SBTTL  READ AST FROM READING ATTACH REQUEST RESPONSE
                        13BA  2437 ;---
                        13BA  2438 ;
                        13BA  2439 ; THIS AST ROUTINE IS CALLED UPON THE COMPLETION OF READING THE
                        13BA  2440 ; ATTACH REQUEST RESPONSE. IT HANDLES THE PROCESSING OF THE
                        13BA  2441 ; DETACHED FLAG. IT IS DOWN AT AST LEVEL TO INSURE SYNCHRONIZATION
                        13BA  2442 ; BETWEEN THE CURRENT PROCESS AND THE PROCESS TO BE ATTACHED.
                        13BA  2443 ; THE ADDRESS OF A QUAD WORD IS PASSED AS THE AST PARAMETER. THE QUAD WORD
                        13BA  2444 ; CONTAINS THE INPUT BUFFER FROM THE READ COMMAND (1ST LONG WORD) AND
                        13BA  2445 ; THE ADDRESS OF THE PRC TABLE AREA (2ND LONG WORD).
                        13BA  2446 ;
                        13BA  2447 ; INPUTS:
                        13BA  2448 ;
                        13BA  2449 ;       4(SP) = AST PARAMETER
                        13BA  2450 ;
                        13BA  2451 ; OUTPUTS:
                        13BA  2452 ;
                        13BA  2453 ;       PRC_V_DETACHED BIT IN PRC_W_FLAGS SET IF INPUT BUFFER = 1
                        13BA  2454 ;       CLEARED OTHERWISE
                        13BA  2455 ;
                        13BA  2456 ;---
                        13BA  2457
                        13BA  2458 READ_AST:
                  0804  13BA  2459          .WORD   ^M<R2,R11>
                        13BC  2460
     52   04 AC    D0  13BC  2461          MOVL    4(AP),R2                ; GET ADDR OF QUAD WORD BUFFER
     5B   04 A2    D0  13C0  2462          MOVL    4(R2),R11               ; GET ADDR. OF CLI PROCESS WORK AREA
          05 62    E9  13C4  2463          BLBC    (R2),10$                ; DON'T SET DETACHED BIT IF ATTACH REFUSED
                        13C7  2464          SETBIT  PRC_V_DETACHED,PRC_W_FLAGS(R11)  ; MARK CURRENT PROC. DETACHED
                        13CC  2465
                    04  13CC  2466 10$:    RET
```

```
                        13CD  2468          .SBTTL  ATTACH REQUEST AST FROM ANOTHER PROCESS
                        13CD  2469 ;---
                        13CD  2470 ;
                        13CD  2471 ; THIS AST ROUTINE HANDLES AN ATTACH REQUEST FROM ANOTHER PROCESS
                        13CD  2472 ; THROUGH THE ATTACH MAILBOX ASSOCIATED WITH THIS PROCESS.
                        13CD  2473 ;
                        13CD  2474 ; INPUTS:
                        13CD  2475 ;
                        13CD  2476 ;       4(AP) = ADDRESS OF PRC AREA
                        13CD  2477 ;
                        13CD  2478 ; OUTPUTS:
                        13CD  2479 ;
                        13CD  2480 ;       NONE
                        13CD  2481 ;---
                        13CD  2482
                        13CD  2483 ATTACH_AST:
                  087C  13CD  2484          .WORD   ^M<R2,R3,R4,R5,R6,R11>
                        13CF  2485
          5B    04 AC   D0  13CF  2486          MOVL    4(AP),R11                       ; GET ADDRESS OF CLI PROCESS WORK AREA
          56    F0 AE   9E  13D3  2487          MOVAB   -ATTMBX_MAXMSG(SP),R6           ; ALLOCATE RECORD BUFFER
             5E   56      D0  13D7  2488          MOVL    R6,SP                           ; REMOVE IT FROM THE STACK
                        13DA  2489
                        13DA  2490 ;
                        13DA  2491 ; IF HANGUP AST IS PENDING, THEN WAKEUP PROCESS SO WE CAN TERMINATE
                        13DA  2492 ;
       76 68 AB   0C    E0  13DA  2493          BBS     #PRC_V_HANGUP,PRC_W_FLAGS(R11),80$ ; WAKE UP IF HANGUP PENDING
                        13DF  2494
                        13DF  2495 ;
                        13DF  2496 ; GET THE ATTACH REQUEST MESSAGE FROM THE MAILBOX.  IF THERE IS MORE THAN
                        13DF  2497 ; ONE MESSAGE IN THE MAILBOX, ONLY HONOR THE LAST ONE.
                        13DF  2498 ;
             53   D4    13DF  2499          CLRL    R3                              ; ASSUME NO MESSAGE READ
             7E   7C    13E1  2500          CLRQ    -(SP)                           ; ALLOCATE AN IOSB
          52    5E   D0  13E3  2501          MOVL    SP,R2                           ; POINT TO IT
                        13E6  2502
                        13E6  2503 10$:     $QIOW_S FUNC=#IO$_READVBLK!IO$M_NOW,-   ; READ THE ATTACH MAILBOX
                        13E6  2504                  CHAN=PRC_W_ATTMBX(R11),-
                        13E6  2505                  EFN=#EXE$C_SYSEFN,-
                        13E6  2506                  IOSB=(R2),-
                        13E6  2507                  P1=(R6),P2=#ATTMBX_MAXMSG
          09 50   E9  1408  2508          BLBC    R0,20$                          ; BRANCH IF ERROR
          06 62   E9  140B  2509          BLBC    (R2),20$                        ; BRANCH IF ERROR (EOF)
       53 02 A2   32  140E  2510          CVTWL   2(R2),R3                        ; SAVE LENGTH OF LAST MESSAGE
             D2   11  1412  2511          BRB     10$                             ; LOOP UNTIL MAILBOX CLEANED OUT
             53   D5  1414  2512 20$:     TSTL    R3                              ; ANY MESSAGE READ?
             48   13  1416  2513          BEQL    90$                             ; IF NOT, IGNORE ATTENTION AST
             54   D4  1418  2514          CLRL    R4                              ; ASSUME THE ANSWER IS "NO"
    02 68 AB   0F    E1  141A  2515          BBC     #PRC_V_DETACHED,PRC_W_FLAGS(R11),30$ ; IF NOT DETACHED, SAY NO
             54   D6  141F  2516          INCL    R4                              ; ELSE, RESPOND WITH YES
          52    5E   D0  1421  2517 30$:     MOVL    SP,R2                           ; RESTORE ADDRESS OF IOSB
          66    54   D0  1424  2518          MOVL    R4,(R6)                         ; MOVE RECORD INTO RECORD BUFFER
                        1427  2519          $QIOW_S FUNC=#IO$_WRITEVBLK,-           ; WRITE YES/NO RESPONSE TO MAILBOX
                        1427  2520                  CHAN=PRC_W_ATTMBX(R11),-
                        1427  2521                  EFN=#EXE$C_SYSEFN,-
                        1427  2522                  IOSB=(R2),-
                        1427  2523                  P1=(R6),P2=#4                   ; ONLY A LONGWORD
          16 50   E9  1447  2524          BLBC    R0,90$                          ; BRANCH IF ERROR DETECTED
```

```
                13 62   E9  144A  2525          BLBC    (R2),90$                    ; BRANCH IF ERROR DETECTED
                10 54   E9  144D  2526          BLBC    R4,90$                      ; IF ANSWER WAS NO, THEN EXIT
                        1450  2527              CLRBIT  PRC_V_DETACHED,PRC_W_FLAGS(R11) ; MARK NO LONGER DETACHED
                        1455  2528  80$:        $WAKE_S                             ; WAKE UP CURRENT PROCESS
                        04    1460  2529  90$:  RET
```

```
                     1461  2531          .SBTTL  SUBPROCESS TERMINATION AST ROUTINE
                     1461  2532 ;---
                     1461  2533 ;
                     1461  2534 ; THIS AST IS CALLED WHEN A MESSAGE IS WRITTEN INTO THE TERMINATION
                     1461  2535 ; MAILBOX, INDICATING THAT A SUBPROCESS HAS GONE AWAY.
                     1461  2536 ;
                     1461  2537 ; INPUTS:
                     1461  2538 ;
                     1461  2539 ;       4(AP) = ADDRESS OF TMBX AREA
                     1461  2540 ;
                     1461  2541 ; OUTPUTS:
                     1461  2542 ;
                     1461  2543 ;       NONE
                     1461  2544 ;---
                     1461  2545
          000000D3   1461  2546 LGI$_FACILITY = 211                     ; LOGINOUT FACILITY CODE
                     1461  2547
                     1461  2548 TERMINATION_AST:
              09EC   1461  2549          .WORD   ^M<R2,R3,R5,R6,R7,R8,R11>
                     1463  2550
                     1463  2551 ;
                     1463  2552 ; GET THE ADDRESS OF THE TMBX DATA STRUCTURE FROM THE AST ARGUMENT LIST.
                     1463  2553 ; LOOK IN IT FOR THE ADDRESS OF THE PRC DATA STRUCTURE AND THE TERMINATION
                     1463  2554 ; MAILBOX CHANNEL NUMBER.
                     1463  2555 ;
    57   04 AC   D0  1463  2556          MOVL    4(AP),R7                 ; GET ADDRESS OF TMBX BLOCK
    5B   0C A7   D0  1467  2557          MOVL    TMBX_L_PRC(R7),R11       ; GET ADDRESS OF CLI PROCESS WORK AR
                     146B  2558
                     146B  2559 ;
                     146B  2560 ; READ THE TERMINATION MAILBOX.  IF NO MESSAGES ARE LEFT, THEN WE ARE READY TO
                     146B  2561 ; RESET OR DELETE THE MAILBOX AND RETURN FROM THE AST.
                     146B  2562 ;
    5E   A4 AE   9E  146B  2563          MOVAB   -8-ACC$C_TERMLEN(SP),SP  ; ALLOCATE BUFFER SPACE FOR
    58   5E   D0    146F  2564 10$:      MOVL    SP,R8                    ;    THE IOSB + RECORD BUFFER
                     1472  2565          $QIOW_S FUNC=#IO$_READVBLK!IO$M_NOW,- ; READ THE TERMINATION MAILBOX
                     1472  2566                  CHAN=TMBX_W_CHANNEL(R7),-
                     1472  2567                  EFN=#EXE$C_SYSEFN,-
                     1472  2568                  IOSB=(R8),-
                     1472  2569                  P1=8(R8),-
                     1472  2570                  P2=#ACC$C_TERMLEN
      03 50   E9     1499  2571          BLBC    R0,15$                   ; BRANCH IF ERROR
      03 68   E8     149C  2572          BLBS    (R8),20$                 ; BRANCH IF MESSAGE READ
       00E2   31     149F  2573 15$:      BRW     200$                     ; MAILBOX CLEANED OUT - DONE
                     14A2  2574
                     14A2  2575 ;
                     14A2  2576 ; SEARCH THE LIST OF OUTSTANDING SUBPROCESS CONTEXT BLOCKS LOOKING FOR
                     14A2  2577 ; A MATCH WITH THE SUBPROCESS WHICH JUST TERMINATED.
                     14A2  2578 ;
                     14A2  2579          ASSUME  SPWN_L_LINK EQ 0
    55   00C0 CB   9E  14A2  2580 20$:      MOVAB   PRC_L_SPWN(R11),R5       ; GET ADDRESS OF SPWN BLOCK LISTHEAD
    56   65   D0      14A7  2581 25$:      MOVL    SPWN_L_LINK(R5),R6       ; GET ADDRESS OF NEXT ENTRY IN LIST
         C3   13      14AA  2582          BEQL    10$                      ; IF NOT IN LIST, IGNORE MESSAGE
      10 A8   D1      14AC  2583          CMPL    ACC$L_PID+8(R8),-        ; DOES PID MATCH?
      40 A6           14AF  2584                  SPWN_L_SUBPID(R6)
         05   13      14B1  2585          BEQL    30$                      ; BRANCH IF SO
    55   56   D0      14B3  2586          MOVL    R6,R5                    ; SKIP TO NEXT ENTRY IN LIST
         EF   11      14B6  2587          BRB     25$                      ; KEEP LOOPING
```

```
                        14B8  2588
                        14B8  2589 ;
                        14B8  2590 ; THE PROCESS WHICH JUST TERMINATED HAS BEEN FOUND IN THE LIST OF
                        14B8  2591 ; CURRENTLY ACTIVE SPAWNED PROCESSES.  TAKE ANY TERMINATION ACTIONS WHICH
                        14B8  2592 ; ARE APPROPRIATE.
                        14B8  2593 ;
                        14B8  2594 ; SET THE TERMINATION STATUS IN THE SPWN BLOCK.
                        14B8  2595 ;
      0C A8    D0       14B8  2596 30$:    MOVL    ACC$L_FINALSTS+8(R8),-          ; SAVE THE TERMINATION STATUS
      44 A6             14BB  2597                 SPWN_L_STATUS(R6)               ;
           04    12     14BD  2598                 BNEQ    50$                     ; BRANCH IF STATUS IS NON-ZERO
  44 A6    01    D0     14BF  2599                 MOVL    #1,SPWN_L_STATUS(R6)    ; IF "NO STATUS", RETURN SUCCESS
                        14C3  2600
                        14C3  2601 ;
                        14C3  2602 ; GET CURRENT IMAGE COUNT.  IF NOT THE SAME AS THE IMAGE COUNT AT THE TIME
                        14C3  2603 ; OF SUBPROCESS CREATION, THEN DO NOT EVEN CONSIDER QUEUEING A TERMINATION,
                        14C3  2604 ; SETTING A TERMINATION EVENT FLAG, OR SETTING TERMINATION STATUS.
                        14C3  2605 ;
           7E    7C     14C3  2606 50$:    CLRQ    -(SP)                           ; CREATE GETJPI ITEM LIST
        F8 AE    9F     14C5  2607         PUSHAB  -2*4(SP)                        ; SET BUFFER ADDRESS
041A0004 8F    DD       14C8  2608         PUSHL   #JPI$_IMAGECOUNT@16+4           ; REQUEST IMAGE COUNT, SET BUFFER LE
     50    5E    D0     14CE  2609         MOVL    SP,R0                           ; SET ADDRESS OF ITEM LIST
                        14D1  2610         $GETJPIW_S  ITMLST=(R0),-               ; GET PROCESS IMAGE COUNT
                        14D1  2611                     IOSB=(R8),-                 ;
                        14D1  2612                     EFN=#EXE$C_SYSEFN           ;
           51 8ED0      14E8  2613         POPL    R1                              ; GET THE CURRENT IMAGE COUNT
     5E    0C    C0     14EB  2614         ADDL    #3*4,SP                         ; CLEANUP STACK
  5C A6    51    D1     14EE  2615         CMPL    R1,SPWN_L_IMAGCNT(R6)           ; SAME IMAGE AS ISSUED SPAWN?
           37    12     14F2  2616         BNEQ    80$                             ; NO, THEN DO NOT QUEUE AST OR SET E
                        14F4  2617
                        14F4  2618 ;
                        14F4  2619 ; IF USER REQUESTED THAT THE TERMINATION STATUS BE RETURNED, THEN TRY TO RETURN
                        14F4  2620 ; IT NOW.
  50    54 A6    D0     14F4  2621         MOVL    SPWN_L_STSADR(R6),R0            ; SHOULD STATUS BE RETURNED TO CALLE
           0D    13     14F8  2622         BEQL    60$                             ; BRANCH IF NOT
                        14FA  2623         IFNOWRT #4,@SPWN_L_STSADR(R6),60$,-     ; SKIP IF NOT WRITABLE
                        14FA  2624                 SPWN_B_ACMODE(R6)               ;
      44 A6    D0     1502  2625         MOVL    SPWN_L_STATUS(R6),-             ; RETURN STATUS TO CALLER
      54 B6           1505  2626                 @SPWN_L_STSADR(R6)              ;
                        1507  2627
                        1507  2628 ;
                        1507  2629 ; IF A TERMINATION AST WAS REQUESTED, THEN QUEUE IT UP NOW.
                        1507  2630 ;
      4C A6    D5     1507  2631 60$:    TSTL    SPWN_L_ASTADR(R6)               ; AST ROUTINE REQUESTED?
           10    13     150A  2632         BEQL    70$                             ; BRANCH IF NOT
                        150C  2633         $DCLAST_S ASTADR=@SPWN_L_ASTADR(R6),-   ; QUEUE THE AST
                        150C  2634                 ASTPRM=SPWN_L_ASTPRM(R6),-      ;
                        150C  2635                 ACMODE=SPWN_B_ACMODE(R6)        ;
                        151C  2636
                        151C  2637 ;
                        151C  2638 ; IF A TERMINATION EF WAS REQUESTED, THEN SET IT NOW.
                        151C  2639 ;
      0F A6    95     151C  2640 70$:    TSTB    SPWN_B_EFN(R6)                  ; EVENT FLAG REQUESTED?
           0A    19     151F  2641         BLSS    80$                             ; BRANCH IF NOT
                        1521  2642         $SETEF_S EFN=SPWN_B_EFN(R6)             ; SET THE EVENT FLAG
                        152B  2643
                        152B  2644 ;
```

```
                     152B  2645 ; PREVENT THE PARENT FROM HANGING TRYING TO SEND ADDITIONAL MESSAGES TO THE
                     152B  2646 ; CONTEXT MAILBOX BY CANCELING AND DEASSIGNING THE CHANNEL TO THAT MAILBOX.
                     152B  2647 ;
                     152B  2648 80$:    $CANCEL_S CHAN=SPWN_W_CHAN(R6)          ; STOP ANY CURRENT WRITES TO MAILBOX
                     1536  2649         $DASSGN_S CHAN=SPWN_W_CHAN(R6)          ; PREVENT FURTHER WRITES TO MAILBOX
          0A A6   B4 1541  2650         CLRW    SPWN_W_CHAN(R6)                 ; INDICATE CHANNEL WAS "REMOVED"
                     1544  2651
                     1544  2652 ;
                     1544  2653 ; IF /NOTIFY SPECIFIED, THEN BROADCAST NOTIFICATION MESSAGE.
                     1544  2654 ;
            08   E1 1544  2655 90$:    BBC     #SPWN_V_NOTIFY,-                ; SKIP IF NOT /NOTIFY
       03 0C A6      1546  2656                 SPWN_W_FLAGS(R6),95$            ;
             0067 30 1549  2657         BSBW    NOTIFY_MESSAGE                  ; BROADCAST NOTIFICATON MESSAGE
                     154C  2658
                     154C  2659 ;
                     154C  2660 ; DECREMENT TERMINATION MAILBOX REFERENCE COUNT.
                     154C  2661 ;
          08 A7   97 154C  2662 95$:    DECB    TMBX_B_REFS(R7)                 ; DECREMENT REFERENCE COUNT TO MAILB
                     154F  2663
                     154F  2664 ;
                     154F  2665 ; DELETE THE WRITE MAILBOX IF ALL THROUGH WITH IT.
                     154F  2666 ;
            0B   E0 154F  2667         BBS     #SPWN_V_OUTPUT,-                ; SKIP IF NO MAILBOX
       03 0C A6      1551  2668                 SPWN_W_FLAGS(R6),97$            ;
             FDBB 30 1554  2669         BSBW    DELETE_OUTMBX                   ; UPDATE OUTPUT MBX USAGE
                     1557  2670
                     1557  2671 ;
                     1557  2672 ; IF SUBPROCESS WAS CREATED /WAIT, THEN ASSUME THAT THE SUBPROCESS THAT JUST
                     1557  2673 ; DIED WAS ATTACHED TO THE TERMINAL AND IS NOW LOGGING ITSELF OUT.  ALSO,
                     1557  2674 ; ASSUME THAT THE PARENT PROCESS WAS DETACHED FROM THE TERMINAL AND SHOULD
                     1557  2675 ; NOW BE AWAKENED TO RECEIVE CONTROL.
                     1557  2676 ;
                     1557  2677 ; NOTE THAT THESE ASSUMPTIONS ARE NOT VALID IN THE CASE OF AN "ATTACHED"
                     1557  2678 ; COUSIN DELETING ITS "DETACHED" COUSIN.
                     1557  2679 ;
                     1557  2680 ; ALSO MARK THE SUBPROCESS INACTIVE SO THAT THE PARENT WILL DELETE THIS
                     1557  2681 ; SPWN BLOCK.
                     1557  2682 ;
            02   E1 1557  2683 97$:    BBC     #SPWN_V_WAIT,-                  ; SKIP IF /NOWAIT
       18 0C A6      1559  2684                 SPWN_W_FLAGS(R6),100$           ;
                     155C  2685         CLRBIT  PRC_V_DETACHED,PRC_W_FLAGS(R11) ; MARK NO LONGER DETACHED
                     1561  2686         CLRBIT  SPWN_V_ACTIVE,SPWN_W_FLAGS(R6)  ; MARK SUBPROCESS INACTIVE
                     1566  2687         $WAKE_S                                 ; WAKE UP CURRENT PROCESS
             FEFB 31 1571  2688         BRW     10$                             ; GET NEXT MESSAGE FROM MAILBOX
                     1574  2689
                     1574  2690 ;
                     1574  2691 ; IF THE SUBPROCESS WAS CREATED /NOWAIT, THEN REMOVE THE SPWN BLOCK FROM THE
                     1574  2692 ; LIST OF OUTSTANDING PROCESSES, AND DEALLOCATE IT.
                     1574  2693 ;
                     1574  2694 100$:   ASSUME  SPWN_L_LINK EQ 0                ;
          65   66   D0 1574  2695         MOVL    (R6),(R5)                       ; REMOVE FROM LINKED LIST NOW THAT
                     1577  2696                                                 ; THE PROCESS IS NO LONGER ACTIVE
          50   56   D0 1577  2697         MOVL    R6,R0                           ; SET ADDRESS OF BLOCK
       51    04 A6   3C 157A  2698         MOVZWL  SPWN_W_SIZE(R6),R1              ; SET LENGTH OF BLOCK
             EA7F' 30 157E  2699         BSBW    DCL$DEADYNMEM                   ; DEALLOCATE SPWN BLOCK
             FEEB 31 1581  2700         BRW     10$                             ; LOOP UNTIL MAILBOX CLEANED OUT
                     1584  2701
```

```
                     1584  2702 ;
                     1584  2703 ; IF REF COUNT FOR THIS MAILBOX IS NOW ZERO, THEN DEASSIGN IT AND REMOVE
                     1584  2704 ; THE TMBX BLOCK FROM THE LINKED LIST.  OTHERWISE, RE-ENABLE WRITE ATTENTION
                     1584  2705 ; AST ON THE MAILBOX.
                     1584  2706 ;
08 A7   95           1584  2707 200$:    TSTB    TMBX_B_REFS(R7)                 ; TEST REFERENCE COUNT TO MAILBOX
   05   12           1587  2708           BNEQ    210$                            ; BRANCH IF STILL OUTSTANDING USES
 F873   30           1589  2709           BSBW    DELETE_TMBX                     ; DELETE TERMINATION MAILBOX AND TMB
   24   11           158C  2710           BRB     220$                            ; DO NOT BOTHER RESETING THE AST
                     158E  2711 210$:     $QIOW_S FUNC=#IO$_SETMODE!IO$M_WRTATTN,- ; RESET ATTENTION AST ON MAILBOX
                     158E  2712                   CHAN=TMBX_W_CHANNEL(R7),-       ;
                     158E  2713                   IOSB=(R8),-                     ;
                     158E  2714                   EFN=#EXE$C_SYSEFN,-             ;
                     158E  2715                   P1=TERMINATION_AST,-            ; ADDRESS OF AST ROUTINE
                     158E  2716                   P2=R7                           ; PASS ADDRESS OF TMBX BLOCK
   04                15B2  2717 220$:     RET                                     ;
                     15B3  2718
```

```
                        15B3  2720          .SBTTL  BROADCAST NOTIFICATION MESSAGE
                        15B3  2721 ;---
                        15B3  2722 ;
                        15B3  2723 ; THIS ROUTINE BROADCASTS A MESSAGE INDICATING THAT A /NOWAIT SUBPROCESS HAS
                        15B3  2724 ; TERMINATED.
                        15B3  2725 ;
                        15B3  2726 ; INPUTS:
                        15B3  2727 ;
                        15B3  2728 ;       NONE
                        15B3  2729 ;
                        15B3  2730 ; OUTPUTS:
                        15B3  2731 ;
                        15B3  2732 ;       NONE
                        15B3  2733 ;---
                        15B3  2734
                        15B3  2735 NOTIFY_MESSAGE:
          5E   2C   C2  15B3  2736          SUBL    #NOTIFY_LEN,SP                  ; ALLOCATE BUFFER ON THE STACK
          7E   5E   D0  15B6  2737          MOVL    SP,-(SP)                        ; BUILD BUFFER DESCRIPTOR
          7E   2C   D0  15B9  2738          MOVL    #NOTIFY_LEN,-(SP)               ;
          53   5E   D0  15BC  2739          MOVL    SP,R3                           ; SAVE ADDRESS OF DESCRIPTOR
     52   0092 C6   9E  15BF  2740          MOVAB   SPWN_T_PROCESS(R6),R2           ; GET PROCESS NAME
     51   EAB1 CF   9E  15C4  2741          MOVAB   NOTIFY_MSG,R1                   ; GET ADDRESS OF ASCIC MSG
          50   81   9A  15C9  2742          MOVZBL  (R1)+,R0                        ; GET LENGTH OF MSG
          7E   50   7D  15CC  2743          MOVQ    R0,-(SP)                        ; PUSH DESCRIPTOR ON STACK
          51   5E   D0  15CF  2744          MOVL    SP,R1                           ; SAVE ADDRESS OF DESCRIPTOR
                        15D2  2745          $FAO_S  CTRSTR=(R1),-                   ; NOTIFICATION MESSAGE
                        15D2  2746                  OUTLEN=(R3),-                   ; ADDR OF RESULT LENGTH
                        15D2  2747                  OUTBUF=(R3),-                   ; ADDR OF BUFFER DESC
                        15D2  2748                  P1=R2                           ; ASCIC SUBPROCESS NAME ADDR
51  00000028'8F   D0    15E1  2749          MOVL    #CTL$AG_CLIDATA+PPD$T_INPDVI,R1 ; GET ADDRESS OF ASCIC DEVICE NAME
          50   81   9A  15E8  2750          MOVZBL  (R1)+,R0                        ; GET LENGTH OF DEVICE
          6E   50   7D  15EB  2751          MOVQ    R0,(SP)                         ; PUT DESCRIPTOR ON STACK
          52   5E   D0  15EE  2752          MOVL    SP,R2                           ; SAVE ADDRESS OF DESCRIPTOR
               7E   7C  15F1  2753          CLRQ    -(SP)                           ; ALLOCATE AN IOSB
          50   6E   D0  15F3  2754          MOVL    (SP),R0                         ;
                        15F6  2755
                        15F6  2756          $BRKTHRUW_S     MSGBUF=(R3),-           ; BROADCAST THE MESSAGE
                        15F6  2757                          SENDTO=(R2),-           ;
                        15F6  2758                          SNDTYP=#BRK$C_DEVICE,-  ;
                        15F6  2759                          REQID=#BRK$C_DCL,-      ;
                        15F6  2760                          EFN=#31,-               ;
                        15F6  2761                          IOSB=(R0)               ;
                        1611  2762
5E  00000044 8F   C0    1611  2763          ADDL    #8+8+8+NOTIFY_LEN,SP            ; RESTORE STACK
                  05    1618  2764          RSB
                        1619  2765
                        1619  2766          .END
```

```
$$.TMP1                 =  00000001
$$.TMP2                 =  00000064
$$T1                    =  00000000
$$T2                    =  00000004
ACC$C_TERMLEN           =  00000054
ACC$L_FINALSTS          =  00000004
ACC$L_PID               =  00000008
ATTACH_AST                 000013CD  R      02
ATTACH_NAME                0000001C  R      02
ATTMBX_MAXMSG           =  00000010
BRK$C_DCL               =  00000006
BRK$C_DEVICE            =  00000001
CHECK_FOR_HANGUP           000011BD  R      02
CHECK_TRMMBX               00000884  R      02
CLI$K_ATTA_IDEN            ********    X    02
CLI$K_SPAW_CARR            ********    X    02
CLI$K_SPAW_CLI             ********    X    02
CLI$K_SPAW_INPU            ********    X    02
CLI$K_SPAW_KEYP            ********    X    02
CLI$K_SPAW_LOG             ********    X    02
CLI$K_SPAW_LOGI            ********    X    02
CLI$K_SPAW_NOTI            ********    X    02
CLI$K_SPAW_OUTP            ********    X    02
CLI$K_SPAW_PROC            ********    X    02
CLI$K_SPAW_PROM            ********    X    02
CLI$K_SPAW_SYMB            ********    X    02
CLI$K_SPAW_TABL            ********    X    02
CLI$K_SPAW_WAIT            ********    X    02
CLI$_ATTACHED           =  0003FD09
CLI$_BUFOVF             =  00038018
CLI$_EXPSYN             =  00038038
CLI$_INVFILSPE          =  00038200
CLI$_NORMAL             =  00030001
CLI$_NOTIFY             =  00038250
CLI$_REFUSED            =  000388C2
CLI$_RETURNED           =  0003FD11
CLI$_SPAWNED            =  0003FD01
CLI$_SPWNIO             =  000388EA
CLI$_STRTOOLNG          =  000388FA
CLI$_SYMTOOLNG          =  00038218
CLI$_TRMMBX             =  000388F2
CLI_NAME                   0000006A  R      02
COM                        00000060  R      02
CONSTRUCT_PRCNAM           000004D8  R      02
CREATE_ATTMBX              00000CE2  R      02
CREATE_OUTMBX              00001208  R      02
CREATE_TMBX                00000D71  R      02
CTL$AG_CLIDATA             ********    X    02
CTL$GL_LNMDIRECT           ********    X    02
CTL$GL_LNMHASH             ********    X    02
CTL$GT_CLINAME             ********    X    02
CTL$GT_SPAWNCLI            ********    X    02
CTL$GT_SPAWNTABLE          ********    X    02
CTL$GT_TABLENAME           ********    X    02
CTX_B_ACMODE               00000004
CTX_B_CONTINUE             00000012
CTX_B_FLAGS                0000000E
CTX_B_KEYLENGTH                       00000002
CTX_B_NFLAGS                          00000005
CTX_B_NONUNIQUE                       00000006
CTX_B_PROMPTLEN                       0000000F
CTX_B_SYMTAB                          00000004
CTX_B_SYMTYPE                         00000005
CTX_B_TFLAGS                          00000005
CTX_B_TRANCNT                         00000006
CTX_C_BINARY                    =     00000002
CTX_C_CLISYM                    =     00000003
CTX_C_CMDSTR                    =     00000001
CTX_C_GLOBAL                    =     00000000
CTX_C_HDRLEN                          00000033
CTX_C_HEADER                    =     00000000
CTX_C_KEYPAD                    =     00000004
CTX_C_KEYSTATE                  =     00000004
CTX_C_KEYTABL                   =     00000002
CTX_C_LNMNAME                   =     00000006
CTX_C_LNMTABLE                  =     00000005
CTX_C_LOCAL                     =     00000001
CTX_C_MAXLEN                    =     00000400
CTX_C_PERM                      =     00000001
CTX_C_STRING                    =     00000000
CTX_G_PROMPT                          00000013
CTX_K_HDRLEN                          00000033
CTX_L_OUTOFBAND                       0000000A
CTX_L_QUOTA                           00000008
CTX_Q_PROCPRIV                        00000002
CTX_T_CMDSTR                          00000002
CTX_T_KEYSTATE                        00000003
CTX_T_LNMNAME                         00000007
CTX_T_LNMTABLE                        0000000C
CTX_T_LOGNAM                          00000005
CTX_T_SYMBOL                          00000007
CTX_V_AUTOLOGO                  =     00000000
CTX_V_MODE                      =     00000001
CTX_V_VERIFY                    =     00000002
CTX_V_VERIMAGE                  =     00000003
CTX_V_WAIT                      =     00000004
CTX_W_ENTSIZE                         00000002
CTX_W_PMPTCTRL                        00000010
CTX_W_PROT                            00000006
CTX_W_TYPE                            00000000
DC$_TERM                        =     00000042
DCL$ABORT                             ********    X    02
DCL$ALLDYNMEM                         ********    X    02
DCL$ATTACH                            00000A01  RG       02
DCL$ATTACH2                           00000A4F  RG       02
DCL$CNVASCBIN                         ********    X    02
DCL$CRLF                              ********    X    02
DCL$C_PROMPTLEN                       ********    X    02
DCL$DEADYNMEM                         ********    X    02
DCL$FORMMSG                           ********    X    02
DCL$GETDVAL                           ********    X    02
DCL$GETNVAL                           ********    X    02
DCL$RESETOOB                          ********    X    02
DCL$RESTART                           ********    X    02
```

```
DCL$SPAWN                       00000098 RG    02
DCL$SPAWN2                      0000025F RG    02
DCL$SPAWNOUT                    ******** X     02
DEALLOC_SPWN                    00000E19 R     02
DELETE_ATTMBX                   00000D62 R     02
DELETE_OUTMBX                   00001312 R     02
DELETE_TMBX                     00000DFF R     02
DEV$V_REC                   =   00000000
DEV$V_TRM                   =   00000002
DIB$W_UNIT                  =   0000000C
DVI$_DEVCLASS               =   00000004
DVI$_DEVNAM                 =   00000020
ENT_K_MAX_PROMPT            =   00000020
EXE$C_SYSEFN                    ******** X     02
FAB$B_BID                   =   00000000
FAB$B_BLN                   =   00000001
FAB$B_DNS                   =   00000035
FAB$B_FNS                   =   00000034
FAB$C_BID                   =   00000003
FAB$C_BLN                   =   00000050
FAB$L_DEV                   =   00000040
FAB$L_DNA                   =   00000030
FAB$L_FNA                   =   0000002C
FAB$L_FOP                   =   00000004
FAB$L_NAM                   =   00000028
FAB$V_PPF                   =   00000012
GET_DEVICE                      000011C6 R     02
IO$M_NOW                    =   00000040
IO$M_TT_PROCESS             =   00002000
IO$M_WRTATTN                =   00000100
IO$_READVBLK                =   00000031
IO$_SENSEMODE               =   00000027
IO$_SETMODE                 =   00000023
IO$_WRITEOF                 =   00000028
IO$_WRITEVBLK               =   00000030
JPI$_ASTLM                  =   00000409
JPI$_BIOLM                  =   00000310
JPI$_DFWSCNT                =   00000403
JPI$_DIOLM                  =   00000313
JPI$_IMAGECOUNT             =   0000041A
JPI$_MASTER_PID             =   00000325
JPI$_PID                    =   00000319
JPI$_PRCNAM                 =   0000031C
JPI$_PRIB                   =   00000309
JPI$_PROCPRIV               =   00000204
JPI$_USERNAME               =   00000202
JPI$_WSEXTENT               =   00000416
JPI$_WSQUOTA                =   00000402
LGI$_FACILITY               =   000000D3
LNM$C_NAMLENGTH             =   000000FF
LNMB$B_ACMODE               =   0000000B
LNMB$B_FLAGS                =   00000010
LNMB$L_TABLE                =   0000000C
LNMB$M_TABLE                =   00000008
LNMB$T_NAME                 =   00000011
LNMB$V_CONFINE              =   00000001
LNMB$V_TABLE                =   00000003
LNMHSH$C_BUCKET             =   0000000C
LNMHSH$L_MASK               =   00000000
LNMTH$L_BYTESLM             =   0000001D
LNMTH$L_CHILD               =   00000011
LNMTH$L_NAME                =   00000009
LNMTH$L_PARENT              =   0000000D
LNMTH$L_SIBLING             =   00000015
LNMX$B_FLAGS                =   00000000
LNMX$T_XLATION              =   00000004
LNMX$V_XEND                 =   00000002
LOG                             00000065 R     02
LOGINOUT                        00000000 R     02
NAM$B_BID                   =   00000000
NAM$B_BLN                   =   00000001
NAM$B_DEV                   =   00000039
NAM$B_ESL                   =   0000000B
NAM$B_ESS                   =   0000000A
NAM$B_NOP                   =   00000008
NAM$C_BID                   =   00000002
NAM$C_BLN                   =   00000060
NAM$C_MAXRSS                =   000000FF
NAM$L_DEV                   =   00000044
NAM$L_ESA                   =   0000000C
NAM$L_FNB                   =   00000034
NAM$M_NOCONCEAL             =   00000010
NAM$V_PPF                   =   00000010
NL                              0000005B R     02
NOTIFY_LEN                  =   0000002C
NOTIFY_MESSAGE                  000015B3 R     02
NOTIFY_MSG                      00000079 R     02
OUTPUT_NAME                     0000002B R     02
PPD$B_NPROCS                    0000001C
PPD$C_LENGTH                    00000168
PPD$K_LENGTH                    00000168
PPD$L_INPDEV                    00000044
PPD$L_LGI                       00000014
PPD$L_LSTSTATUS                 00000018
PPD$L_OUTDEV                    00000064
PPD$L_PRC                       00000008
PPD$Q_CLIREG                    00000004
PPD$Q_CLISYMTBL                 0000000C
PPD$T_FILENAME                  00000068
PPD$T_INPDVI                    00000028
PPD$T_OUTDVI                    00000048
PPD$W_FLAGS                     00000002
PPD$W_INPCHAN                   0000001E
PPD$W_INPDID                    0000003E
PPD$W_INPFID                    00000038
PPD$W_INPIFI                    00000020
PPD$W_INPISI                    00000022
PPD$W_OUTDID                    0000005E
PPD$W_OUTFID                    00000058
PPD$W_OUTIFI                    00000024
PPD$W_OUTISI                    00000026
PPD$W_SIZE                      00000000
PQL$_ASTLM                  =   00000001
PQL$_BIOLM                  =   00000002
```

```
PQL$_DIOLM                = 00000005
PQL$_LISTEND              = 00000000
PQL$_WSDEFAULT            = 0000000B
PQL$_WSEXTENT             = 0000000D
PQL$_WSQUOTA              = 0000000A
PRC$M_CLISPEC             = 00001000
PRC$M_INTER               = 00000400
PRCNAM_NAME                 00000014 R     02
PRC_B_CONTINUE              000000F3
PRC_B_DEFRADIX              000000AE
PRC_B_EXMDEPMOD             000000AD
PRC_B_EXMDEPWID             000000AC
PRC_B_EXONLYL               0000012D
PRC_B_FLAGS2                000000AF
PRC_B_IMGFLAG               00000078
PRC_B_OUTFLAGS              0000012C
PRC_B_PROMPTLEN             000000F0
PRC_C_LENGTH                00000534
PRC_G_COMMANDS              00000133
PRC_G_PROMPT                000000F4
PRC_K_HEX                 = 00000000
PRC_K_LENGTH                00000534
PRC_L_CURRKEY               00000048
PRC_L_EXMDEPADR             000000A8
PRC_L_EXTARG                00000094
PRC_L_EXTBLK                0000008C
PRC_L_EXTCOD                0000009C
PRC_L_EXTHND                00000090
PRC_L_EXTPRM                00000098
PRC_L_IDFLNK                000000BC
PRC_L_IMGACTSTS             00000080
PRC_L_INDCLOCK              0000007C
PRC_L_INDEPTH               0000005C
PRC_L_INDFAB                0000001C
PRC_L_INDINPRAB             00000014
PRC_L_INDOUTRAB             00000018
PRC_L_INPRAB                00000008
PRC_L_LASTKEY               0000004C
PRC_L_LSTSTATUS             000000B0
PRC_L_ONCTLY                000000B8
PRC_L_ONERROR               0000006C
PRC_L_OUTOFBAND             000000B4
PRC_L_OUTRAB                0000000C
PRC_L_OUTRABCTX             00000118
PRC_L_PPFLIST               00000070
PRC_L_RECALLPTR             0000012F
PRC_L_RESTART               00000058
PRC_L_SAVAP                 00000000
PRC_L_SAVFP                 00000004
PRC_L_SEVERITY              00000050
PRC_L_SPWN                  000000C0
PRC_L_STACKLM               000000A4
PRC_L_STACKPT               000000A0
PRC_L_STATUS                00000054
PRC_L_STS                   00000084
PRC_L_STV                   00000088
PRC_L_SYMBOL                00000060
PRC_L_TMBX                  00000074
PRC_L_TRMLIST               00000010
PRC_Q_ALLOCREG              00000020
PRC_Q_COMMAND               000000E0
PRC_Q_FLUSHTIME             000000D0
PRC_Q_GLOBAL                00000028
PRC_Q_IMAGENAME             000000D8
PRC_Q_KEYPAD                00000040
PRC_Q_LABEL                 00000030
PRC_Q_LOCAL                 00000038
PRC_Q_SAVEPRIV              000000E8
PRC_T_OUTDVI                0000011C
PRC_V_DETACHED            = 0000000F
PRC_V_HANGUP              = 0000000C
PRC_V_MODE                = 00000006
PRC_V_VERIFY              = 00000007
PRC_V_VERIMAGE            = 00000007
PRC_V_YLEVEL              = 0000000B
PRC_W_ASTIOSB               000000C6
PRC_W_ASTRETN               000000C8
PRC_W_ASTSTATUS             000000C4
PRC_W_ATTMBX                0000007A
PRC_W_FLAGS                 00000068
PRC_W_INPCHAN               00000064
PRC_W_ONLEVEL               0000006A
PRC_W_OUTIFI                00000114
PRC_W_OUTISI                00000116
PRC_W_OUTMBXCHN             000000CA
PRC_W_OUTMBXREF             000000CE
PRC_W_OUTMBXSIZ             000000CC
PRC_W_PMPTCTRL              000000F1
PRC_W_WAITIOSB              00000066
PSL$C_EXEC                = 00000001
PSL$S_PRVMOD              = 00000002
PSL$V_PRVMOD              = 00000016
PTR_B_LEVEL                 00000004
PTR_B_NUMBER                00000005
PTR_B_PARMCNT               00000006
PTR_B_VALUE                 00000000
PTR_C_LENGTH                0000000C
PTR_K_COLON               = 00000002
PTR_K_COMDQUAL            = 00000000
PTR_K_LENGTH                0000000C
PTR_K_PARAMETR            = 00000003
PTR_L_DESCR                 00000000
PTR_L_ENTITY                00000008
PTR_V_FLAGS               = 00000014
PTR_V_NEGATE              = 00000014
READ_AST                    000013BA R     02
RESTORE_CONTEXT             000006F1 R     02
RETURNED_MESSAGE            00000E36 R     02
SPAWN_EXIT                  000006C2 R     02
SPAWN_PROCESS               00000544 R     02
SPWN_B_ACMODE               0000000E
SPWN_B_CONTINUE             000000A5
SPWN_B_EFN                  0000000F
SPWN_B_PROMPTLEN            000000A2
```

```
SPWN_C_LENGTH                   000000D6
SPWN_G_PROMPT                   000000A6
SPWN_G_QUOTAS                   00000060
SPWN_K_LENGTH                   000000D6
SPWN_L_ASTADR                   0000004C
SPWN_L_ASTPRM                   00000050
SPWN_L_IMAGCNT                  0000005C
SPWN_L_LINK                     00000000
SPWN_L_OUTOFBAND                00000058
SPWN_L_PRIB                     00000048
SPWN_L_STATUS                   00000044
SPWN_L_STSADR                   00000054
SPWN_L_SUBPID                   00000040
SPWN_M_CLI                    = 00002000
SPWN_M_CLISYM                 = 00000020
SPWN_M_KEYPAD                 = 00001000
SPWN_M_LOG                    = 00000001
SPWN_M_LOGNAM                 = 00000040
SPWN_M_TABLE                  = 00008000
SPWN_M_WAIT                   = 00000004
SPWN_Q_CLI                      000000C6
SPWN_Q_CMDSTR                   00000030
SPWN_Q_INPUT                    00000020
SPWN_Q_IOSB                     00000038
SPWN_Q_MBXNAM                   00000010
SPWN_Q_OUTPUT                   00000028
SPWN_Q_PRCNAM                   00000018
SPWN_Q_TABLE                    000000CE
SPWN_T_PROCESS                  00000092
SPWN_V_ACTIVE                 = 00000007
SPWN_V_AUTOLOGO               = 00000003
SPWN_V_CLI                    = 0000000D
SPWN_V_CLISYM                 = 00000005
SPWN_V_INPUT                  = 0000000A
SPWN_V_KEYPAD                 = 0000000C
SPWN_V_LOG                    = 00000000
SPWN_V_LOGNAM                 = 00000006
SPWN_V_MODE                   = 00000004
SPWN_V_NOCTX                  = 0000000E
SPWN_V_NOTIFY                 = 00000008
SPWN_V_OUTPUT                 = 0000000B
SPWN_V_PRCNAM                 = 00000001
SPWN_V_PROMPT                 = 00000009
SPWN_V_TABLE                  = 0000000F
SPWN_V_WAIT                   = 00000002
SPWN_W_CHAN                     0000000A
SPWN_W_FLAGS                    0000000C
SPWN_W_PMPTCTRL                 000000A3
SPWN_W_SIZE                     00000004
SPWN_W_UNIT                     00000008
SS$_DUPLNAM                   = 00000094
SS$_ENDOFFILE                 = 00000870
SYI$_MAXBUF                   = 0000104F
SYM_B_FLAGS                     0000000B
SYM_B_NONUNIQUE                 0000000B
SYM_B_TYPE                      0000000A
SYM_K_BINARY                  = 00000002
SYM_K_KEYPAD                  = 00000004
SYM_K_PERM                    = 00000001
SYM_K_STRING                  = 00000000
SYM_L_BL                        00000004
SYM_L_FL                        00000000
SYM_T_SYMBOL                    0000000C
SYM_W_SIZE                      00000008
SYS$ASSIGN                      ********  GX    02
SYS$BRKTHRUW                    ********  GX    02
SYS$CANCEL                      ********  GX    02
SYS$CLREF                       ********  GX    02
SYS$CREMBX                      ********  GX    02
SYS$CREPRC                      ********  GX    02
SYS$DASSGN                      ********  GX    02
SYS$DCLAST                      ********  GX    02
SYS$DELPRC                      ********  GX    02
SYS$FAO                         ********   X    02
SYS$GETCHN                      ********  GX    02
SYS$GETDVIW                     ********  GX    02
SYS$GETJPIW                     ********  GX    02
SYS$GETSYIW                     ********  GX    02
SYS$HIBER                       ********  GX    02
SYS$INPUT                       0000003A  R     02
SYS$OUTPUT                      00000044  R     02
SYS$PARSE                       ********  GX    02
SYS$QIOW                        ********  GX    02
SYS$SEARCH                      ********  GX    02
SYS$SETEF                       ********  GX    02
SYS$SYSTEM                      0000004F  R     02
SYS$WAKE                        ********  GX    02
TERMINATION_AST                 00001461  R     02
TMBX_B_REFS                     00000008
TMBX_C_LENGTH                   00000010
TMBX_C_MAXREFS                = 00000004
TMBX_K_LENGTH                   00000010
TMBX_L_LINK                     00000000
TMBX_L_PRC                      0000000C
TMBX_W_CHANNEL                  00000006
TMBX_W_SIZE                     0000000A
TMBX_W_UNIT                     00000004
TT2$V_DCL_MAILBX              = 00000009
VERIFY_INPUT                    00000710  R     02
VERIFY_OUTPUT                   000008EF  R     02
WRITE_AST                       00001329  R     02
WRITE_CONTEXT                   00000E80  R     02
WRITE_MAILBOX                   0000111B  R     02
WRITE_SYMBOLS                   00001149  R     02
WRK_B_CMDOPT                    FFFFFFC3
WRK_B_MAXPARM                   FFFFFFD0
WRK_B_MINPARM                   FFFFFFD1
WRK_B_PARMCNT                   FFFFFFCE
WRK_B_PARMSUM                   FFFFFFCF
WRK_B_RECALLCNT                 FFFFFFC5
WRK_B_VALLEV                    FFFFFFC4
WRK_B_VERBTYP                   FFFFFFC2
WRK_C_INPBUFSIZ               = 00000100
WRK_C_LENGTH                    FFFFF486
```

```
WRK_G_BUFFER                    FFFFF492
WRK_G_INPBUF                    FFFFF896
WRK_G_RESULT                    FFFFF9B6
WRK_K_LENGTH                    FFFFF486
WRK_L_CHARPTR                   FFFFF48E
WRK_L_DISALLOW                  FFFFFFE6
WRK_L_ERRORRTN                  FFFFF9AE
WRK_L_EXPANDPTR                 FFFFF486
WRK_L_IMAGE                     FFFFFFE2
WRK_L_MARKPTR                   FFFFF48A
WRK_L_PAROUT                    FFFFFFD2
WRK_L_PMPTADDR                  FFFFF9A2
WRK_L_PROMPTRTN                 FFFFF9A6
WRK_L_PROPTR                    FFFFFFC6
WRK_L_QUABLK                    FFFFFFCA
WRK_L_READRTN                   FFFFF9AA
WRK_L_RECALLPTR                 FFFFFFEA
WRK_L_RSLEND                    FFFFFFB6
WRK_L_RSLNXT                    FFFFFFBA
WRK_L_SAVAP                     FFFFFFF8
WRK_L_SAVFP                     FFFFFFFC
WRK_L_SAVSP                     FFFFFFF4
WRK_L_SIGNALRTN                 FFFFFFD6
WRK_L_SPECRTN                   FFFFF9B2
WRK_L_TAB_VEC                   FFFFFFDE
WRK_L_VERB                      FFFFFFBE
WRK_W_FLAGS                     FFFFFFF0
WRK_W_FLAGS2                    FFFFFFF2
WRK_W_IMGCHAN                   FFFFFFEE
WRK_W_PMPTLEN                   FFFFF99E
_$$_                        = 000000EF
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DCL$ZCODE	00001619 (5657.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	16	00:00:00.05	00:00:01.03
Command processing	97	00:00:00.61	00:00:07.31
Pass 1	835	00:00:43.45	00:02:15.40
Symbol table sort	0	00:00:05.26	00:00:14.71
Pass 2	399	00:00:09.91	00:00:32.77
Symbol table output	0	00:00:00.40	00:00:00.95
Psect synopsis output	0	00:00:00.02	00:00:00.38
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1347	00:00:59.71	00:03:12.55

```
The working set limit was 2700 pages.
240218 bytes (470 pages) of virtual memory were used to buffer the intermediate code.
There were 180 pages of symbol table space allocated to hold 3272 non-local and 181 local symbols.
2766 source lines were read in Pass 1, producing 32 object records in Pass 2.
96 pages of virtual memory were used to define 72 macros.
```

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
_255DUA28:[SYSLIB]SYSBLDMLB.MLB;1	0
_255DUA28:[DCL.OBJ]DCL.MLB;1	14
_255DUA28:[SYS.OBJ]LIB.MLB;1	3
_255DUA28:[SYSLIB]STARLET.MLB;2	46
TOTALS (all libraries)	63

```
3762 GETS were required to define 63 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SPAWN/OBJ=OBJ$:SPAWN MSRC$:SPAWN/UPDATE=(ENH$:SPAWN)+EXECML$/LIB+LIB$:DCL/LIB+SYS$LIBRARY:SYSBLDMLB/LIB
```

SHOW
LIS

RPDCL
LIS

SET
LIS

SPAWN
LIS

RPSUB
LIS

PREFIX
REQ

STACLINT
LIS

STATUS
LIS

ANALYZE
LIS

DCXMSG
LIS

STASTUB
LIS

DCXDEF
MDL

EXPAND
LIS

SYSOUTPUT
LIS

DCXPRVDEF
MDL

STATEMENT
LIS

DCX

COMPRESS
LIS

TRANSFER
LIS

SYMBOL
LIS

DCXSHR
MAP

SUBS
LIS